4

0300

00169.002292

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	:	Examiner: NYA
PETER WILLIAM MITCHELL ILBERY	)	
	:	Group Art Unit: NYA
Application No.: 10/029,267	)	
	:	
Filed: December 28, 2001	)	
	:	
For: ERROR DIFFUSION USING NEXT SCANLINE ERROR IMPULSE RESPONSE	)	
	:	
	)	February 5, 2002

Commissioner for Patents
Washington, D.C. 20231

SUBMISSION OF PRIORITY DOCUMENT

Sir:

In support of Applicant's claim for priority under 35 U.S.C. § 119, enclosed is a certified copy of the following foreign application:

PR2347, filed December 29, 2000.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,

Attorney for Applicant

Registration No. 42,476

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

NY_MAIN 235694 v 1

10/029,267

Patent Office
Canberra

I, JONNE YABSLEY, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PR 2347 for a patent by CANON KABUSHIKI KAISHA filed on 29 December 2000.

WITNESS my hand this
Tenth day of January 2002

JR Yabsley

JONNE YABSLEY
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF
PRIORITY DOCUMENT

S&F Ref: 528985

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Error Diffusion Using Next Scanline Error Impulse Response

Name and Address of Applicant:

Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, Japan

Name of Inventor:

Peter William Mitchell Ilbery

This invention is best described in the following statement:

ERROR DIFFUSION USING NEXT SCANLINE ERROR IMPULSE RESPONSE

Copyright Notice

Technical Field of the Invention

The present invention relates to the field of digital image processing and more particularly to apparatus and method for digital halftoning continuous tone images.

Background Art

Digital images

An image is typically represented digitally as a rectangular array of pixels with each pixel having one of a restricted set of legitimate pixel values. Digital images may be black and white in which case the restricted set of legitimate pixel values is used to encode an optical density or luminance score. Digital images may also be colour where the restricted set of pixel values may encode an optical density or intensity score for each of a number of colour channels - for example Cyan, Magenta, Yellow and Black (CMYK) or Red, Green and Blue (RGB).

Digital images are common where the image value per pixel per colour channel is an 8 bit unsigned number - providing intensity values in the range 0 through 255. Such images are often called "continuous tone" images because of the reasonably large number (256) of legitimate intensity values.

Digital halftoning

By contrast, digital images are often printed on devices which provide a more limited variation in intensity or colour representation per pixel. For example, many Bubble Jet printers only provide the ability to print or not print a dot of each of a Cyan, Magenta, Yellow and Black ink at each pixel position.

In order to print a digital image on a printer with lower colour resolution than the digital image, it is necessary to use the original image to generate an image with the required lower colour resolution such that the generated image has a similar appearance to the original image. This process of generating a digital image of similar appearance where each pixel colour value is within a smaller set of legitimate pixel colour values, is known as digital halftoning.

For ease of explanation, digital halftoning is hereafter described for the case where the input image is a single colour channel, 8 bit per pixel image and the halftoned image is a 1 bit per pixel (bi-level) image. The input image values are known as greyscale values or grey levels. Extensions of digital halftoning from this monochrome bi-level case to cases where the halftoned image pixels have more than 2 legitimate output values (multi-level halftoning) and extensions to digital halftoning of colour images can be performed. See for example, "Digital Halftoning" Ulichney R., MIT Press, 1987, pp 340-342.

Consider the case of a single colour channel, 8 bit per pixel, image which is halftoned to generate a 1 bit per pixel (bi-level) image which is printable on a black and white bubble jet printer, for which each pixel of the halftoned image has one of 2 legitimate pixel values, a "no dot" value and a "dot" value.

An image region in the halftoned image will print as a minimum optical density ("fully white") region when all pixels of the region have the "no dot" value corresponding to non-placement of an ink dot; it will print as a maximum optical density ("fully black")

region when all pixels of the region have the "dot" value corresponding to placement of an ink dot; and it will print as an intermediate optical density (halftone) region when some of the pixels of the region have the "no dot" value and some of the pixels have the "dot" value.

A highlight (near white) region in the halftoned image will have only a few of the pixels with the "dot" value - in these regions the "dot" halftone output value is the minority or exceptional result. A shadow (near black) region will have only a few of the pixels with the "no dot" value - in these regions the "no dot" halftone output value is the minority or exceptional result.

The role of the halftoning process is to generate the printable image so that an appropriate number of ink dots will print in appropriate pixel positions so that there is a good match between the optical density of image regions in the original image and the average optical density of the matching image regions in the printed image.

Error diffusion

Error diffusion is a digital halftoning method originally developed by Floyd and Steinberg and described in the publication "An Adaptive Algorithm for Spatial Greyscale", Proceedings of the SID 17/2, pp 75-77 (1976). Error diffusion as developed by Floyd and Steinberg is hereafter described as "standard error diffusion".

An overview of standard error diffusion is now provided.

The standard error diffusion algorithm processes pixels line by line from the top of the image to the bottom of the image. Each line (or "scanline") is processed one pixel at a time from left to right.

The standard error diffusion algorithm employs a pixel decision rule in which a modified input image pixel value is compared against a threshold value. If the pixel's modified input value is less than the threshold, then the pixel's halftone output value is

assigned to be the lower halftone output value; and if it is greater, then the pixel's halftone output value is assigned to be the higher halftone output value.

Following determination of the pixel halftone output value, an error is determined for the pixel as the difference between the pixel's modified input value and the pixel's halftone output value. The error is distributed to neighbouring, as yet unprocessed, pixels according to a set of weighting coefficients.

A pixel's modified input value is the sum of the pixel's input image value and a neighbourhood error value for the pixel. The neighbourhood error for a pixel is the sum of the errors distributed to that pixel from previously processed neighbouring pixels.

The set of weighting coefficients is known as an error diffusion mask. Each weighting coefficient of the error diffusion mask is associated with a pixel offset. When a pixel is processed, the error distributed from the pixel to an as yet unprocessed pixel is the pixel error multiplied by the weight corresponding to the offset from the pixel to the unprocessed pixel.

Note that the error at a pixel which is distributed to its neighbours can be considered as a sum of the neighbourhood error at the pixel and the pixel-only error being the pixel's input value less the pixel's halftone output value.

The error diffusion mask described by Floyd and Steinberg is shown in Fig. 1. The error at a pixel is distributed in the proportions 7/16, 1/16, 5/16 and 3/16 to 4 neighbouring pixels which are as yet unprocessed as indicated in the diagram. Fig. 1 shows an error diffusion mask 100 in which an error for a current pixel 108 is distributed in the proportions indicated to four neighbouring pixels eg. 106 in the fractional proportions shown. Previously processed pixels eg. 110 are shown using a shaded representation, and it is noted that the current pixel 108 lies on a current scanline 102.

Previously processed pixels lie above a bold line 104 in Fig. 1, and a scanline processing direction is depicted by a horizontal arrow 112.

It is noted that the sum of the weighting coefficients is 1. As a result 100% of a pixel's error is transferred to its neighbouring pixels. If the sum of weighting coefficients were greater than 1, then error would be amplified and could build up without bound. If the sum of weighting coefficients were less than 1, then error would be reduced. By having the sum of weighting coefficients equal to unity, the average intensity of a region of the halftoned image tends to match the average intensity of that region in the input image which is a very desirable characteristic of a halftoning process.

In implementations of standard error diffusion where multiplication of error by a weighting coefficient results in significant rounding errors, it is necessary to co-ordinate the calculation of error distributions to neighbouring pixels so that effectively 100% of current pixel error is transferred on. This can be achieved by determining error distributions to all but one of the neighbouring pixels by multiplication by a weight and determining the error distribution to the remaining neighbouring pixel by subtracting the other error distributions from the total error to be distributed.

For improved execution speed, error distributions corresponding to a particular pixel error value are often determined in advance and retrieved from a look up table.

In the Floyd and Steinberg error diffusion mask, the pixels which receive error distributions from a current pixel are on the current scanline and succeeding scanline only. An implementation of error diffusion with this mask requires the use of a single "line store" memory to store neighbourhood error values. The memory is referred to as a "line store" because it is required to store a neighbourhood error value for each pixel of a scanline. Many of the modifications to error diffusion which are referred to below are achieved at the cost of extra memory for an additional one or more line stores.

Error diffusion algorithms are used in the printing and display of digital images. Many modifications to the standard error diffusion algorithm have been developed.

Worm artifacts

The error diffusion algorithm suffers from the disadvantage that in image regions of very low or very high intensity, it generates a pattern of image values in the halftoned image which are poorly spread - the exceptional values are concentrated in wavy lines. These patterns can be very noticeable and distracting to a viewer of the image - they are often known as "worm" artifacts.

Fig. 5 shows a section of a halftone output image generated by Floyd Steinberg error diffusion which shows worm artifacts. This halftone output was generated by bi-level halftoning of an 8 bit per pixel monochrome source image with a constant grey value of 253. Fig. 5 shows a halftone output image 500 which, as noted, shows worm artifacts, as illustrated by, for example, pairs of pixels 502, and a quadruplet of pixels 504.

Modifications to error diffusion have been developed to reduce worm artifacts.

Prior art methods of reducing worm artifacts

One method of reducing worm artifacts is by addition of some randomisation. The randomisation may be achieved by adding noise to the input image, by adding noise to the thresholds or by randomising the error distribution to neighbouring pixels. A large amount of noise or randomisation can be added to fully avoid worm artifacts; however, this also seriously degrades the halftoned image.

Another method of reducing worm artifacts is to vary the direction in which scanlines are processed. By way of example, U.S. Patent No. 4,955,065 titled "System for Producing Dithered Images from Continuous-tone Image Data" to Ulichney discloses

error diffusion with perturbed weighting coefficients and bi-directional scanline processing.

Another method of reducing worm artifacts is by use of larger error diffusion masks. For example, in "Error Diffusion Algorithm with Reduced Artifacts", Eschbach R., Proceedings of the IS&T's 45th Annual Conference, May 10-15, 1992, either a large or a small error diffusion mask is used depending on the input image grey level. The large error diffusion mask is suited to use in image regions of very high or very low greyscale, reducing the worm artifacts in those regions. A disadvantage of this method is that large error diffusion masks which distribute error to pixels of more than 1 succeeding scanline require additional error line stores and associated processing.

A further method of reducing worm artifacts by use of an "extended distribution set" style of error diffusion mask is described in U.S. Patent No. 5,353,127, titled "Method for Quantization Gray Level Pixel Data with Extended Distribution Set" to Shiau and Fan. This patent describes error diffusion masks including pixel positions of only the current and succeeding scanline with additional pixel positions on the next scanline to the left of (that is, behind) the current pixel.

Fig. 2 shows an error diffusion mask of U.S. Patent No. 5,353,127. An error for a current pixel 208 is, in this case, distributed to five neighbouring pixels eg. 206, where each of these five pixels receives a fractional proportion of the error as indicated. Scanline processing is from left to right as depicted by an arrow 212, and previously processed pixels, eg. 210, are full shaded and lie above a bold line 204. The current pixel 208 lines on a current scanline 202. While this method is successful at reducing worm artifacts and only requires a single error line store, worm artifacts are still evident for bi-level halftoning of 8 bit grey scale image data, as can be seen in Fig. 6.

Fig. 6 shows example halftone output generated by bi-level error diffusion, using the mask of fig. 2, for an 8 bit per pixel source image with constant grey value of 253. Fig. 6 shows an image 600 which exhibits worm artifacts as exemplified by reference numerals 602 and 604.

Prior art methods of preventing artifacts

While the above methods are successful in reducing worm artifacts, complete prevention of worm artifacts can also be achieved.

One method of preventing worm artifacts is by modulating threshold values. U.S. Patent No. 5,535,019 titled "Error diffusion halftoning with homogeneous response in high/low intensity image regions" to Eschbach discloses a modification to error diffusion which adjusts the error diffusion threshold according to the halftone output and according to the input intensity using a threshold impulse function, for the purpose of preventing worm artifacts.

Another modification to error diffusion which prevents worm artifacts by threshold modulation is described in "A serpentine error diffusion kernel with threshold modulation for homogeneous dot distribution", Hong D., Kim C., Japan Hardcopy '98 pp 363-366, 1998 which is also published in IS&T's Recent Progress In Digital Halftoning II (1999) pp 306-309.

Another method of preventing worm artifacts is the addition of grey level dependent periodic noise to the input image, described in "Modified error diffusion with smoothly dispersed dots in highlight and shadow", Japan Hardcopy '98 pp 379-382, 1998 which is also published in IS&T's Recent Progress In Digital Halftoning II (1999) pp 310-313.

Another method of preventing worm artifacts is by imposing output position constraints. An example of this method is provided in "An error diffusion algorithm with

output constraints for homogeneous highlight and shadow dot distribution", Marcu G., Proceedings of SPIE, Vol 3300, pp 341-352 (1998).

Disadvantages of prior art methods of preventing worm artifacts

In many cases the desirability of a halftoning algorithm is determined by how fast it executes and how easy it is to implement. For example, in software implementations in a printer driver on a general purpose computer, the algorithm execution speed is very important; whereas in special purpose hardware, the algorithm complexity and memory usage are very important because they relate strongly to the expense of the circuitry.

Use of additional line store memory by a halftoning algorithm generally indicates that it will execute slower in software and is more expensive to implement in hardware.

The modification to error diffusion disclosed in US Patent No. 5,535,019 to Eschbach requires use of memory for an additional line store to store threshold adjustment values generated by preceding scanlines for use by a current scanline. The modification also requires additional processing including addition of threshold impulse values and dampening of threshold values transferred to subsequent scanlines.

The modification to error diffusion described in the previously mentioned paper titled "A serpentine error diffusion kernel with threshold modulation for homogeneous dot distribution" also requires use of an additional line store memory to store threshold adjustment values. Additional processing is also required to diffuse threshold adjustment values.

The modification to error diffusion described in the previously mentioned paper titled "Modified error diffusion with smoothly dispersed dots in highlight and shadow" requires memory for additional line stores to store several neighbouring scanlines of

processed input image data from which filtered input image values for a current scanline are determined; the filtered input image values are used in turn to determine the noise to be added to the input image data.

The modification to error diffusion described in the previously mentioned paper titled "An error diffusion algorithm with output constraints for homogeneous highlight and shadow dot distribution" requires memory for additional lines stores to store halftoned image data for several previously processed scanlines. This modification also includes processing to exclude minority halftone output results when that result is present in a certain portion of the previously processed scanlines.

In summary, all the modifications listed above which prevent worm artifacts in error diffusion require use of additional line store memory together with the processing associated with use of that additional memory.

References made to prior art documents in the present description in no way constitutes an acknowledgment that the prior art documents are part of the common general knowledge.

Summary of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of halftoning an image, said method comprising steps of:

determining an output value of a current pixel on a current scanline using a sum of an input value for the current pixel and a neighbourhood error value at the current pixel;

determining an error at the current pixel as the difference between (i) the sum of the input value for the current pixel and the neighbourhood error value at the current pixel, and (ii) the output value of the current pixel;

adding a proportion of the error at the current pixel to neighbourhood error values at as yet unprocessed pixels of a subsequent scanline in accordance with a next scanline error impulse response; wherein said next scanline error impulse response:

approximates a function which spreads with self-convolution in proportion to the degree of self-convolution.

According to a second aspect of the invention, there is provided a method of generating an error diffusion mask suitable for use with any one of the aforementioned methods.

According to a second aspect of the invention, there is provided an error diffusion mask suitable for use with any of the aforementioned methods.

According to another aspect of the invention, there is provided an apparatus for implementing any one of the aforementioned methods.

According to another aspect of the invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for implementing any one of the methods described above.

Other aspects of the invention are also disclosed.

Brief Description of the Drawings

One or more embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 shows an error diffusion mask as described by Floyd and Steinberg;

Fig. 5 shows a halftone output image generated in accordance with the Floyd Steinberg mask shown in Fig.1;

Fig. 2 shows an error diffusion mask in accordance with US Patent No. 5,353,127 (Shiau and Fan);

Fig. 6 shows an exemplary halftone output image in accordance with bi-level error diffusion, using the mask shown in Fig. 2;

Fig. 3 shows the mask position of an exemplary error diffusion mask, designed in accordance with an arrangement described herein;

Fig. 4 shows a tabular representation of the mask weight values of the exemplary error diffusion mask shown in Fig. 3;

Fig. 7 shows a halftone output image produced using the mask shown in Fig. 3;

Fig. 8 shows distribution of pixel errors for error diffusion using a single error line store;

Fig. 9 shows error diffusion processing for an arbitrary pixel;

Fig. 10 shows an alternate arrangement of error diffusion processing on a per pixel basis;

Fig. 11 shows error diffusion processing per pixel in accordance with current and next scanline error impulse response functions;

Fig. 12 provides a representation of a conservative vector field;

Fig. 13 shows an nth next scanline error impulse response function using an approximate Cauchy distribution;

Fig. 14 shows self convolutions of a next scanline error impulse response function for Floyd Steinberg error diffusion;

Fig. 15 shows self convolutions of a next scanline error impulse response function for US Patent No. 5,353,127 error diffusion;

Fig. 16 shows self convolutions of a next scanline error impulse response function in accordance with a Cauchy distribution;

Fig. 17 is a block diagram representation of a system for performing pixel processing in accordance with a first arrangement described herein;

Fig. 19 shows an exemplary family of error diffusion masks for a second arrangement using a Cauchy distribution;

Fig. 20 shows a tabular representation of a mapping between grey levels and the masks shown in Figure 19;

Fig. 18 shows a block diagram representation of a system for pixel processing using a second arrangement of the Cauchy distribution; and

Fig. 21 is a schematic block diagram of a general purpose computer upon which arrangements described herein can be practiced.

Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

Some portions of the description which follows are explicitly or implicitly presented in terms of algorithms and symbolic representations of operations on data within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that the above and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as "scanning", "calculating", "determining", "replacing", "generating" "initializing", "outputting", or the like, refer to the action and processes of a computer system, or similar electronic device, that manipulates and transforms data represented as physical (electronic) quantities within the registers and memories of the computer system into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present specification also discloses apparatus for performing the operations of the methods. Such apparatus may be specially constructed for the required purposes, or may comprise a general purpose computer or other device selectively activated or reconfigured by a computer program stored in the computer. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein. Alternatively, the construction of more specialized apparatus to perform the required method steps may be appropriate. The structure of a conventional general purpose computer will appear from the description below.

In addition, the present specification also discloses a computer readable medium comprising a computer program for performing the operations of the methods. The computer readable medium is taken herein to include any transmission medium for communicating the computer program between a source and a designation. The transmission medium may include storage devices such as magnetic or optical disks,

memory chips, or other storage devices suitable for interfacing with a general purpose computer. The transmission medium may also include a hard-wired medium such as exemplified in the Internet system, or wireless medium such as exemplified in the GSM mobile telephone system. The computer program is not intended to be limited to any particular programming language and implementation thereof. It will be appreciated that a variety of programming languages and coding thereof may be used to implement the teachings of the disclosure contained herein.

Modifications to error diffusion are now described, which remove worm artifacts in highlight and shadow regions, by design of the error diffusion mask.

Figs. 3 and 4 show an example of an error diffusion mask, designed as described in this document, which generates bi-level halftone output for 8 bit per pixel source images which is substantially free of worm artifacts; that is, all sparse halftone patterns are well-spread. Fig. 3 shows an error diffusion mask 300 in which error for a current pixel 306 is distributed to neighbouring pixels on a current scanline 314, as well as pixels on a "next" scanline 316. Previously processed pixels are shown in full shade above a bold line 318, and it is noted that a current scanline 314 and a next scanline 316 contain pixels to which the error for the current pixel 306 is distributed. Current scanline pixels which receive error distribution from the current pixel 306 are represented by a group 312, and next line pixels are correspondingly represented by a group 310. Fig. 4 shows a tabular representation of the error distribution weight values for the error diffusion mask, and it is noted that the table 400 is made up of four columns 402 to 408. Columns 402 and 406 represent mask positions for the next and current scanlines respectively, and columns 404 and 408 represent the proportions in which the error from the current pixel are distributed.

The first arrangement is described for the case of halftoning a monochrome 8 bit per pixel input image to a bi-level output image.

A monochrome input image with pixel values $g_{i,j}$ being integers in the range 0 to 255 is used to generate a bi-level halftone output image with pixel values $r_{i,j}$ being the integers 0 and 255.

The halftone output value for each pixel is obtained by error diffusion, using an error diffusion mask which has been specially designed so that the next scanline error impulse response function, corresponding to the mask, approximates a Cauchy distribution which has been sampled and normalised.

An example of a suitable error diffusion mask for the first arrangement is the 12-13 mask described in Figs. 3 and 4. This mask has weights for 12 pixel positions on the current scanline and 13 pixel positions on the next scanline and was prepared as described above. Use of such a specially designed error diffusion mask generates halftone output for which all sparse halftone patterns are well spread and substantially worm-free.

Fig. 7 shows halftone output using this mask for a source image of constant grey level 253. Fig. 7 shows an image 700 which is substantially free from worm artifacts, as exemplified by dots 702 and 704, which are well spaced from their neighbours.

This description includes:

- a definition of the term "next scanline error impulse response" function;
- arguments supporting the proposition that in order for error diffusion with mask positions on the current and next scanlines to be worm-free, it is desirable that the next scanline error impulse response function should spread in proportion to the degree of self-convolution, or stated alternatively, should approximate a Cauchy distribution;

- a method for generating error diffusion masks for which the next scanline error impulse response function is optimised to approximate a function which spreads in proportion to the degree of self-convolution, ie. is optimised to approximate a Cauchy distribution.

It will also be shown that the requirement that the next scanline error impulse response function should spread in proportion to the degree of self-convolution is closely related to the requirement that it should approximate a Cauchy distribution.

Influence of a pixel decision on subsequent pixel decisions

With standard error diffusion, if the modified input value of a pixel is lower than the threshold, then the halftone result is set low and the error distributed to neighbouring pixels acts to increase the likelihood that neighbouring pixels are set high. Conversely if the modified input value of a pixel is higher than the threshold, then the halftone result is set high and the error distributed to neighbouring pixels acts to increase the likelihood that neighbouring pixels are set low.

The proportion of the (non-zero) error of an arbitrary reference pixel which contributes to the neighbourhood error of a subsequently processed pixel of interest, is a measure of a disincentive to assign the pixel of interest the same halftone output result as the reference pixel.

Due to the typically sequential processing of pixels, and because the processing of each pixel includes the distribution (according to the error diffusion mask) of that pixel's error, being the sum of that pixel's neighbourhood error (the error distributed to that pixel) and that pixel's pixel-only error, a pixel contributes a proportion of its error to the neighbourhood error of pixels which are beyond those to which it directly distributes error.

Error diffusion implementable with a single error line store

Error diffusion is now described, where:

(a) pixels are processed one at a time across a scanline or row, either left to right or right to left, and scanlines are processed one after another from the top to the bottom of the image; and

(b) the error diffusion mask includes only pixel positions in the current and succeeding scanline.

Fig. 8 is a diagram indicating, for this class of error diffusion, the pixels to which error is distributed. Fig. 8 shows a representation 800 in which a current pixel 802 is shown on a current scanline 812, where scanline processing is from left to right as depicted by an arrow 806, and where previously processed pixels eg. 804 are full shaded. Current scanline mask positions eg. 808 denote mask values for pixel positions on the current scanline, and are designated $mask_{curr}[i](i>0)$. Next scanline mask positions eg. 810 denote mask values for pixel positions on the next scanline, and are designated $mask_{next}[i]$. Mask positions on the current and next scanlines are identified by a horizontal pixel offer "i". Horizontal pixel offsets are shown in Fig. 8 by circled numbers 816.

Error diffusion processing per pixel

Fig. 9 shows error diffusion processing per pixel, for an arbitrary pixel of column i and scanline j. Fig. 9 shows a block diagram representation 900 where an input image grey level at pixel (i,j) ie. 902 is provided to an addition process 904 which outputs, on a line 906, the sum of the input image grey level ($g_{i,j}$), a total error distributed directly to pixel (i,j) from pixels of the current scanline ($e\_curr\_line_{i,j}$), and a total error distributed directly to the pixel (i,j) from pixels of a previous scanline ($e\_prev\_line_{i,j}$).

The output signal on the line 906 is provided, on a line segment 908, to a threshold process 912 where it is compared against a threshold to obtain a halftone result

for the pixel on a line 914, and subsequently on a line segment 916, the halftone result being designated $r_{i,j}$. The modified input value on the line 906 is also provided on a line segment 910 to an adder process 920 which receives a negative value of the halftone result for the pixel on a line segment 918. An output from the addition process 920 is an error for distribution (designated e_combined $_{i,j}$) on a line 922. This error for distribution is provided on a line 924 to a current scanline error distribution unit (designated $mask_{curr}$) 934 which distributes the combined pixel error $e\_combined_{i,j}$, as shown by an exemplary arrow 932, to unprocessed pixels of the current scanline in a current line error buffer 930. An output from the current line error buffer 930 is provided on a line 928 (the signal being designated as $e\_curr\_line_{i,j}$) and is delivered to an adder process 948. The combined pixel error $e\_combined_{i,j}$ is also provided on a line 926 to a next scanline error distribution unit (designated as $mask_{next}$) 936, which outputs, as exemplified by an arrow 938, corresponding values to pixels of the next scanline which are stored in a line store error buffer 940. The buffer 940 provides an output on a line 942 to an error line store 944. On a line 946, a value is retrieved from the error line store, the value being the total error distributed directly to a pixel (i,j) from pixels of the previous scanline, ie. $e\_prev\_line_{i,j}$.

Operation of the process of Fig. 9 is described by the following 4 steps.

Step 1. Derive a modified input value for the pixel as:

$$g_{ij} + e\_curr\_line_{ij} + e\_prev\_line_{ij} \qquad (30)$$

where: g_{ij} is the input image grey level at pixel (i,j), $e\_curr\_line_{ij}$ is the total error distributed directly to pixel (i,j) from pixels of the current scanline, and $e\_prev\_line_{ij}$ is the total error distributed directly to pixel (i,j) from pixels of the previous scanline;

Step 2. Compare the modified input value against a threshold to obtain the halftone result for the pixel, denoted as r_{ij}

Step 3. Calculate an error for distribution as:

$$e\_combined_{ij} = e_{ij} + e\_curr\_line_{ij} + e\_prev\_line_{ij} \quad (1)$$

where: e_{ij} is the pixel-only error equal to $g_{ij} - r_{ij}$

Step 4. Distribute the combined pixel error, $e\_combined_{ij}$ according to the error diffusion mask to unprocessed pixels of the current scanline and pixels of the next scanline.

In step 1, the total error distributed to a pixel (i,j) from previously processed pixels is referred to as the "neighbourhood error" at pixel (i,j). The neighbourhood error, e_nbr_{ij}, modifies the pixel input value prior to thresholding and is given by:

$$e\_nbr_{ij} = e\_curr\_line_{ij} + e\_prev\_line_{ij} \quad (2)$$

In step 4 error values which are portions of the combined pixel error, $e\_combined_{ij}$ are used to update error sum values in a current line error buffer and in a line store error buffer.

Each error sum value in the current line error buffer is associated with a pixel position on the current scanline ahead of the current pixel - a "future pixel" of the current scanline; that error sum value is the sum of error values distributed directly to that future pixel from processed pixels of the current scanline.

Similarly, each error sum value in the line store error buffer is associated with a pixel position on the next scanline - a "next scanline pixel"; that error sum value is the sum of error values distributed directly to that next scanline pixel from processed pixels of the current scanline.

The number of error sum values in the current line error buffer and in the line store error buffer need only be as large as the number of error diffusion mask positions on the current scanline and next scanline respectively.

In step 4, once the error sum values in the line store error buffer have been updated, one value in the line store error buffer is complete, in that it will receive no further contributions, and it is transferred to the error line store.

The number of error sum values in the error line store needs to be (and need only be) as large as the number of pixels on a scanline - hence the name "line store".

The error sum values in the error line store generated by processing one scanline are used as input in the processing of the next scanline. When each pixel is processed:

- one error line store value is read (and the memory location from which it is read becomes available to store another error line store value)
- error sum values in the line store error buffer are updated, and
- one error line store value is written.

Fig. 10 shows an alternative implementation 1000 of the error diffusion processing per pixel. The input image grey level at pixel (i,j), ie $g_{i,j}$ is provided to an adder process 1004 on a line 1002. The adder process 1004 outputs, on a line 1006, the sum of the input image grey level ($g_{i,j}$), the total error distributed directly to pixel (i,j) from pixels of the current scanline (ie. $e\_curr\_line_{i,j}$), and the total error distributed directly to the pixel (i,j) from pixels of the previous scanline (ie. $e\_prev\_line_{i,j}$). This value on the line 1006 is provided, by a line segment 1008, to a threshold process 1012 for comparison against a threshold, which consequently produces, on a line 1014, the output result for the pixel (ie. $r_{i,j}$) on a line 1016.

An inverse of the halftone result (ie. negative $r_{i,j}$) is provided on a line 1018 to an addition process 1020, along with the modified input value (ie. $g_{i,j} + e\_curr\_line_{i,j} + e\_prev\_line_{i,j}$) on a line 1010. An output from the addition process 1020 is provided on a line 1022, this being the error for distribution $e\_combined_{i,j}$. This error for distribution is provided on a line 1024 to the current

scanline error distribution unit (designated as $mask_{curr}$) 1028, which outputs, as exemplified by an arrow 1030, error distributions to unprocessed pixels of the current scanline in a current line error buffer 1032. The buffer 1032 consequently outputs the total error distributed directly to pixel (i,j) from pixels of the current scanline (ie $e\_curr\_line_{i,j}$) on a line 1034 to an addition process 1036. The error for distribution on the line 1022 is also distributed, on a line 1026, to an error line store 1038. From the error line store a weighted sum of error values is retrieved on a line 1040 to a next scanline error distribution unit (designated as $mask_{next}$) 1042, which distributes the weighted sum, as exemplified by an arrow 1044, to a line store error buffer 1046. The buffer 1046 provides, on a line 1048, the total error distributed directly to the pixel (i,j) from pixels of the previous scanline (ie. $e\_prev\_line_{i,j}$) to the addition process 1036.

The difference between the 2 implementations, is that in the alternative implementation:

(a) the combined error at a pixel is written to a location in the error line store;

(b) the contribution to neighbourhood error by direct distribution from pixels of a previous scanline is determined as a weighted sum of error values retrieved from the error line store. That is, in the alternative implementation, distribution of error to pixels of the next scanline is achieved by gathering error values from the line store, rather than distributing error amongst line store locations.

An analysis is presented below concerning self-convolution of a next scanline error impulse response function. That analysis is described referring to Fig. 9; however the same analysis also applies to the alternative implementation of Fig. 10.

Next scanline error impulse response function for error diffusion implementable with a single error line store

Error diffusion is now considered for the case where 100% of the combined pixel error is distributed to as yet unprocessed pixels (step 4 above), and the fraction of error distributed to the next scanline is considered to be non-zero.

Further, error diffusion is considered for the case where the error diffusion mask weights do not change from pixel to pixel. That is, it is assumed that the processing per pixel is spatially invariant.

The distribution of the combined pixel error of an arbitrary reference pixel (but which is not near the left and right edges of the image) is now considered according to the error diffusion mask. Part of that combined pixel error is distributed to pixels of the next scanline, and the remainder of the combined pixel error is distributed to pixels of the current scanline. This remainder error remains subject to error distribution according to the error diffusion mask as part of the processing of the current scanline. With the complete processing of the current scanline, effectively all of the combined error of the reference pixel is distributed to pixels of the next scanline.

The distribution of combined pixel error of an arbitrary reference pixel of the current scanline to pixels of the next scanline, as a result of the complete processing of the scanline, defines a "next scanline error impulse response" function which is denoted as h_{next}. To be precise, h_{next} is defined as a function which maps integers to real values; where $h_{next}[i]$ is that fraction of the error at a reference pixel which, following the complete processing of the scanline of the reference pixel, is distributed to the pixel on the next scanline horizontally offset i from the reference pixel. This is shown illustratively with reference to Fig. 13.

Self-convolutions of the next scanline error impulse response function

As the domain of the function h_{next} is the set of integers, it can also be considered as a (two-sided) sequence.

The set of input image grey levels, g_{ij}, is considered for a scanline j as a sequence, and the sequence is denoted as g_j.

Similarly:

- the sequence of error sum values, $e\_prev\_line_{ij}$, being sums of errors distributed directly to pixels of scanline j from scanline j-1, is denoted as $e\_prev\_line_j$;
- the sequence of halftone output image values for scanline j is denoted as r_j;
- the sequence of pixel-only errors for scanline j is denoted as e_j.

For the class of error diffusion considered, the processing performed for each scanline, j, can be considered to take as input the following:

$$e\_prev\_line_j,\ g_j \tag{31}$$

and produces as output the following:

$$r_j,\ e_j,\ e\_prev\_line_{j+1} \tag{32}$$

For the implementation of Fig. 9, as part of the processing of scanline j, the sequence of values $e\_prev\_line_j$ are read from the error line store and the sequence of values $e\_prev\_line_{j+1}$ are written to the error line store.

From the definition of the next scanline error impulse response function, the following can be written:

$$e\_prev\_line_{j+1} = (e_j + e\_prev\_line_j) * h_{next} \tag{3}$$

That is, the sequence of sums of error values distributed directly to pixels of scanline j+1 from scanline j can be represented as the sum of the error sequence for scanline j and the sequence of sums of error values distributed directly to pixels of

scanline j from scanline j-1 convolved with the next scanline error impulse response function.

Here the convolution of 2 sequences is denoted by '*' and the sequence formed by the convolution of 2 sequences, f and g is defined as:

$$(f * g)[i] = \Sigma_{k \in Z} f[i-k]\, g[k] \quad (33)$$

where Z is the set of all integers.

Assuming that the next scanline error impulse response function is the same for each scanline, (3) can be applied recursively, and the sequence of sums of error values distributed directly to pixels of a scanline from the previous scanline can be written as a weighted sum of pixel only errors of pixels of preceding scanlines.

That is,

$$e\_prev\_line_j = (e_{j-1} + e\_prev\_line_{j-1}) * h_{next} \quad (33)^*$$

$$= (e_{j-1} + (e_{j-2} + e\_prev\_line_{j-2}) * h_{next}) * h_{next} \quad (34)$$

and so on, giving

$$e\_prev\_line_j = \Sigma_{l<j}\, e_l * h_{next}^{*(j-l)} \quad (4)$$

where h_{next}^{*n} denotes the sequence formed as the convolution of the sequence h_{next} with itself 'n-1' times. So that

$$h_{next}^{*1} = h_{next} \quad (35)$$

$$h_{next}^{*2} = h_{next} * h_{next} \text{ and so on.} \quad (36)$$

The current scanline error impulse response function in terms of the error diffusion mask

The error diffusion mask can be represented using 2 functions which map the integers into real values, $mask_{curr}$ and $mask_{next}$, where:

$mask_{curr}[i]$ is the error diffusion mask weight for that pixel on the current scanline horizontally offset by 'i' from the reference pixel being processed; and

$mask_{next}[i]$ is the error diffusion mask weight for that pixel on the next scanline horizontally offset by 'i' from the reference pixel being processed.

Fig. 8 shows horizontal pixel offsets 816 from a reference pixel, together with mask positions 808 and 810 on the current and next scanline respectively.

Due to the sequential processing of pixels of the current scanline,

$$mask_{curr}[i] = 0 \text{ for } i <= 0. \quad (37)$$

As a consequence of the error diffusion processing per pixel and from considering fig. 9 it is seen that:

$$e\_curr\_line_{i,j} = \Sigma_{k \in Z}\, e\_combined_{i-k,j} \cdot mask_{curr}[k] \quad (38)$$

That is, the corresponding sequences are related by the convolution operation:

$$e\_curr\_line_j = e\_combined_j * mask_{curr} \quad (5)$$

The sequence $e\_combined_j$ is the same as $(e_j + e\_curr\_line_j + e\_prev\_line_j)$, and so:

$$e\_curr\_line_j = (e_j + e\_curr\_line_j + e\_prev\_line_j) * mask_{curr} \quad (6)$$

By repeatedly replacing $e\_curr\_line_j$ in the right hand side of (6) with the entire right hand side of (6), the following relationship results:

$$e\_curr\_line_j = (e_j + e\_prev\_line_j) * (mask_{curr} + mask_{curr} * mask_{curr} + ..) \quad (39)$$

That is,

$$e\_curr\_line_j = (e_j + e\_prev\_line_j) * h_{curr} \quad (7)$$

where h_{curr} is the function (sequence) given by:

$$h_{curr} = mask_{curr} + mask_{curr}^{*2} + mask_{curr}^{*3} + .. \quad (8)$$

h_{curr} is equivalently defined by the recursive definition:

$$h_{curr} = (\delta + h_{curr}) * mask_{curr} \quad (9)$$

where δ is the delta function (sequence) given by:

$$\delta[k] = 1 \text{ for } k = 0 \quad (40)$$

$$\delta[k] = 0 \text{ for } k \neq 0 \tag{41}$$

h_{curr} is referred to as the current scanline error impulse response function.

The next scanline error impulse response function in terms of the error diffusion mask

As is the case for the current scanline error impulse response function, the next scanline error impulse response function is determined by the error diffusion mask.

As a consequence of the error diffusion processing per pixel and from considering Fig. 9 the following relationship can be expressed:

$$e\_prev\_line_{i,j+1} = \Sigma_{k \in Z} e\_combined_{i-k,j} \cdot mask_{next}[k] \tag{42}$$

That is, the corresponding sequences are related by the convolution operation:

$$e\_prev\_line_{j+1} = e\_combined_j * mask_{next} \tag{10}$$

As above, using the fact that the sequence $e\_combined_j$ is the same as $(e_j + e\_curr\_line_j + e\_prev\_line_j)$, the following is seen:

$$e\_prev\_line_{j+1} = (e_j + e\_curr\_line_j + e\_prev\_line_j) * mask_{next} \tag{11}$$

By replacing $e\_curr\_line_j$ in the right hand side of (11) with the right hand side of (7), the following is seen:

$$e\_prev\_line_{j+1} = (e_j + e\_prev\_line_j + (e_j + e\_prev\_line_j) * h_{curr}) * mask_{next} \tag{43}$$

which can be re-written as

$$e\_prev\_line_{j+1} = (e_j + e\_prev\_line_j) * ((\delta + h_{curr}) * mask_{next}) \tag{44}$$

Comparing this with (3), it is seen that that the next scanline error impulse response function is given by:

$$h_{next} = (\delta + h_{curr}) * mask_{next} \tag{12}$$

By substituting the expansion of h_{curr} of (8) into (12) it is seen that

$$h_{next} = mask_{next} * (\delta + mask_{curr} + mask_{curr}{}^{*2} + mask_{curr}{}^{*3} + \ldots) \tag{13}$$

Current and next scanline error impulse response functions for standard error diffusion

To assist an understanding of the above definitions of the current and next scanline error impulse response functions, the current and next scanline error impulse response for standard (Floyd Steinberg) error diffusion are now described.

For Floyd Steinberg error diffusion, the error diffusion mask is defined by the following mask functions:

$$\left.\begin{aligned} &mask_{curr}[i] = 0 \text{ for } i \neq 1 \\ &mask_{curr}[1] = 7/16 \end{aligned}\right\} \tag{45}$$

and

$$\left.\begin{aligned} &mask_{next}[i] = 0 \text{ for } i < -1 \text{ and } i > 1 \\ &mask_{next}[-1] = 3/16 \\ &mask_{next}[0] = 5/16 \\ &mask_{next}[1] = 1/16 \end{aligned}\right\} \tag{46}$$

For Floyd Steinberg error diffusion, the current scanline error impulse response function is given by:

$$\left.\begin{aligned} &h_{curr}[i] = 0 \text{ for } i <= 0 \\ &h_{curr}[i] = (7/16)^i \text{ for } i >= 1 \end{aligned}\right\} \tag{47}$$

which is consistent with (8) and (9).

For Floyd Steinberg error diffusion, the next scanline error impulse response function is given by:

$$\left.\begin{aligned} &h_{next}[i] = 0 \text{ for } i < -1 \\ &h_{next}[-1] = 3/16 \\ &h_{next}[0] = 5/16 + 7/16 * 3/16 \\ &h_{next}[i] = (1/16 + 7/16 * 5/16 + (7/16)^2 * 3/16) * (7/16)^{i-1} \text{ for } i > 0 \end{aligned}\right\} \tag{48}$$

which is consistent with (12).

Description of error diffusion processing in terms of the current and next scanline error impulse response functions

This description provides, firstly, further clarification of the current and next scanline error impulse response functions, and secondly, it is the basis for later discussion

concerning desirable characteristics of the current and next scanline error impulse response functions.

Fig. 11 shows error diffusion processing per pixel in terms of the current and next scanline error impulse response functions, h_{curr} and h_{next}. The input image grey level at pixel (i,j), ie. $g_{i,j}$, is provided, as depicted by an arrow 1102, to an addition process 1104. The addition process 1104 provides, as depicted by a line segment 1106, the sum of the input image grey level ($g_{i,j}$), and the total error distributed directly to the pixel (i,j) from pixels of the previous scanline (ie. $e\_prev\_line_{i,j}$) on a line 1108, to an addition process 1112, and also, on a line 1110, to an addition process 1124. The addition process 1112 provides an output on a line 1114 to a threshold process 1116, which compares the value against a threshold, consequently outputting the halftone result for the pixel $r_{i,j}$ on a line 1118. An inverted value of the halftone result (ie. $-r_{i,j}$) is provided on a line 1122 to the addition process 1124.

The addition process 1124 outputs, on a line 1126, and on a line segment 1128, an aggregate value to a current scanline error impulse response distribution unit 1134 (designated h_{curr}) which provides, as exemplified by an arrow 1134, values distributed to pixels of the current scanline in a buffer 1136. From a buffer 1136 on a line 1138, a value labelled as $e\_curr\_line_{i,j}$ is provided to the addition process 1112. In a similar manner, the signal on the line 1126 is provided, on a line segment 1130, to a next scanline error impulse response distribution unit (designated h_{next}) 1140 which, as exemplified by an arrow 1142, distributes a value in accordance with the next scanline error impulse response h_{next} to pixels of the next scanline in a line store error buffer 1144. The buffer 1144 consequently outputs, on a line 1146, a value to the error line store 1148. From the error line store, a value labelled as $e\_prev\_line_{i,j}$ is retrieved and provided to the addition process 1104.

Now the labels, $e\_curr\_line_{i,j}$ and $e\_prev\_line_{i,j}$ in Fig. 11 are consistent with the previous definitions of these terms. This is because

1. the sequence resulting from applying the current scanline error impulse response to the sequence $e_j + e\_prev\_line_j$ is the same as the sequence resulting from applying the current scanline mask coefficients to the sequence $e\_combined_j$, which can be seen from (5) and (7) follows:

$$e\_curr\_line_j = e\_combined_j * mask_{curr} = (e_j + e\_prev\_line_j) * h_{curr} \quad (49)$$

2. the sequence resulting from applying the next scanline error impulse response to the sequence $e_j + e\_prev\_line_j$ is the same as the sequence resulting from applying the next scanline mask coefficients to the sequence $e\_combined_j$, which can be seen from (3) and (10) as follows:

$$e\_prev\_line_{j+1} = e\_combined_j * mask_{next} = (e_j + e\_prev\_line_j) * h_{next} \quad (50)$$

That is, if h_{curr} and h_{next} are defined in terms of $mask_{curr}$ and $mask_{next}$ according to (8) or (9) and (12) or (13), then the processing described in relation to Fig. 11 is equivalent to the processing described in relation to Figs. 9 and 10 in that they produce the same output. In each case, for each pixel, the same quantity, $g_{i,j} + e\_curr\_line_{i,j} + e\_prev\_line_{i,j}$ is applied to the threshold operation.

In general, for implementations of error diffusion, the processing models of Figs. 9 or 10 are preferred over the processing model of Fig. 11. This is because error diffusion masks with non-zero coefficients at a small number of pixel offset positions, correspond to current and next scanline error impulse response functions which typically have an unbounded number of non-zero function values.

However, the current and next scanline error impulse response functions and the processing model of Fig. 11 are useful for describing desirable features of error diffusion

processing, and also for performing experiments to establish desirable features of error diffusion processing.

The neighbourhood error at a current pixel, is the total error distributed directly from previously processed pixels to the current pixel, and is used to modify the current pixel's input grey value prior to thresholding. Considering the sequence of neighbourhood errors for a scanline, the following is seen:

$$e\_nbr_j = e\_curr\_line_j + e\_prev\_line_j \quad (50)^*$$

$$= (e_j + e\_prev\_line_j) * h_{curr} + (e_{j-1} + e\_prev\_line_{j-1}) * h_{next} \quad (51)$$

giving,

$$e\_nbr_j = E_j * h_{curr} + E_{j-1} * h_{next} \quad (14)$$

The above equation (14) neatly isolates the contribution to neighbourhood error from pixels of the current scanline and pixels of the previous scanline in terms of a common error value per pixel. That common error value per pixel warrants a special name. The (previous scanlines) "modified pixel error" for a pixel (i,j) $E_{i,j}$ is defined as:

$$E_{ij} = e_{ij} + e\_prev\_line_{ij} \quad (52)$$

Equation 14 provides assistance in inferring some conclusions regarding desirable characteristics of error diffusion.

Desirable attributes of the next scanline error impulse response function: requirement of unity sum

As noted earlier, in order that the average halftone image output matches the average image input, it is a requirement that the sum of the error diffusion coefficients should be 1. That is,

$$\Sigma_i\, mask_{curr}[i] + \Sigma_i\, mask_{next}[i] = 1 \quad (15)$$

It follows from equations 13 and 15, assuming that the absolute value of the sum of error diffusion mask coefficients on the current scanline is less than 1, that the sum of coefficients of the next scanline error impulse response function is 1:

$$\Sigma_i \, h_{next}[i] = 1 \qquad (16)$$

In summary, for the processing model of Figs. 9 and 10, involving the 2 functions $mask_{curr}$ and $mask_{next}$; the requirement that average halftone image output matches average image input is summarised by (15), whereas for the processing model of Fig. 11, the same requirement is summarised by equation 16, involving only one function h_{next}.

That is, in the processing model of Fig. 11, the current scanline error impulse response function can be varied independently, without compromising the requirement for average halftone output to match average image input.

Desirable attributes of the next scanline error impulse response function: requirement of left - right symmetry

The next scanline error impulse response function as defined by equation 3, and as depicted in Fig. 11, does not show any dependence on whether current scanline processing is left to right or right to left.

If it is assumed that current scanline processing does a very good job of setting halftone output values for the scanline, and consequently generating pixel only error values for the scanline, then it is reasonable to assume that subsequent processing of the next scanline could be left to right or right to left and does not need to compensate for any left-right bias in the pixel only errors of the current scanline. It seems clear then that the next scanline error impulse response function should not be biased to left or right but be left-right symmetric.

A requirement is thus defined that the next scanline error impulse response function should be left-right symmetric.

In fact, some signal shifting or phase distortion due to the current scanline error impulse response function is inevitable because it is a causal filter. However, the signal shifting and phase distortion can be made small. So, it is reasonable to ignore compensation for left-right bias by use of the next scanline error impulse response function, at least for discussion of appropriate spreading of the next scanline error impulse response function.

Desirable attributes of the current and next scanline error impulse response functions: requirement of being monotonic decreasing with increasing horizontal pixel offset

The current scanline error impulse response function measures the degree to which the halftone error at a current pixel on the current scanline should be taken account of in the halftone decision of a subsequently processed pixel on the current scanline.

Also, having concluded, at least to a first approximation, that the next scanline error impulse response function should be left-right symmetric, there is no preference in whether a scanline is processed in the same direction as the preceding scanline. The next scanline error impulse response function contributes to the amount to which the halftone error at a current pixel on the current scanline is taken account of in the halftone decision of a pixel on the next scanline.

It is further suggested that both the current and next scanline impulse response functions should decrease monotonically with increasing horizontal pixel separation.

Desirable attributes of the next scanline error impulse response function: requirement of similar shape of current and next scanline error impulse response functions

Having concluded, at least to a first approximation, that the next scanline error impulse response function should be left-right symmetric, it can be seen using (52) and

(4) that each of the modified pixel error values, E_{ij}, is a weighted sum of pixel-only errors and is obtained by convolution of pixel-only scanline sequences (e_j), with left-right symmetric function $h_{next}*(j-i)$.

Thus, (2) and (14) isolate the asymmetric and symmetric influence of pixel only error of previously processed pixels on the halftone decision at a current pixel.

That is, the term

$$e\_curr\_line_{ij} = (E_j * h_{curr})[i] \tag{53}$$

is an asymmetric contribution to the neighbourhood error of pixel (i,j), being the error distributed to the pixel directly from pixels of the current scanline; whereas the term

$$e\_prev\_line_{ij} = (E_{j-1} * h_{next})[i] \tag{54}$$

is a symmetric contribution, being the error distributed to the pixel directly from pixels of the previous scanline.

In a similar argument to that for inferring monotonicity and left-right symmetry, it is suggested that the contribution to the neighbourhood error of the current pixel, from the modified error of previously processed pixels of the *current* scanline, should reduce with increasing horizontal separation in a similar fashion to the contribution from the modified error of previously processed pixels of the *previous* scanline. In this way no particular horizontal separation is favoured on either the current or previous scanline.

Desirable attributes of the next scanline error impulse response function: requirement of appropriate spreading

Experiments have been performed, observing halftone output when the next scanline error impulse response function has various shapes. In each case, the current scanline response function has the same shape as the next scanline response function, being determined from the next scanline response function by multiplication by a step function. Note that in these simulations the number of non-zero values used for the

current and next scanline response functions was set high to avoid misleading results due to truncation of the functions.

Experiment 1

In this case the next scanline error impulse response function is a sampling of the 2-sided exponential distribution.

$$\left.\begin{aligned} h_{curr}[i] &= a \,.\, w^{i} \text{ for } i >= 1 \\ h_{next}[i] &= ((1 - w) / (1 + w)) \,.\, w^{|i|} \end{aligned}\right\} \tag{55}$$

where a is a positive constant and w is a constant with $0 < w < 1$

Note that with $a = 1$, the above current and next scanline error impulse response functions correspond to the following mask functions

$$\left.\begin{aligned} &mask_{curr}[1] = w \text{ and } mask_{curr}[i] = 0 \text{ for } i \neq 1 \\ &mask_{next}[i] = (1 - w)^2 \; w^{|i|} \text{ for } i <= 0 \text{ and } mask_{next}[i] = 0 \text{ for } i > 0 \end{aligned}\right\} \tag{56}$$

With $a = 1$ and $w = 1/2$, the above mask functions and functions h_{curr} and h_{next} provide an extension to US Patent 5,353, 127 (Shiau & Fan) where the extended distribution set of the error diffusion mask is left extended without limit.

Observations for experiment 1

When $a = 1$ and $w = 1/2$, halftone output at very low or very high grey levels suffers from worm type artifacts where the horizontal separation of minority pixels is too small. These artifacts can be reduced by increasing either of the parameters a or w. However modifying these parameters so that minority pixels at very low or very high grey levels are better separated, introduces other unpleasant artifacts, including artifacts where for other grey levels the vertical separation between pixels becomes too small.

Experiment 2

In this case the next scanline error impulse response function is a sampling of the gaussian distribution.

$$\left.\begin{aligned} h_{curr}[i] &= a \,.\, \exp(-\pi \,.\, (i/b)^2) \text{ for } i >= 1 \\ h_{next}[i] &= c \,.\, \exp(-\pi \,.\, (i/b)^2) \end{aligned}\right\} \quad (57)$$

where a, b, c are positive constants, with c chosen so that $\Sigma_i\, h_{next}[i] = 1$

The parameter b defines the width of the gaussian distribution.

Observations for experiment 2

When a = 1/2 and b = 4, halftone output at low or high grey levels suffers from worm type artifacts where the horizontal separation of minority pixels is too small. These artifacts can be reduced by increasing a or b. However, again, modifying these parameters so that minority pixels at low or high grey levels are better separated, introduces other unpleasant artifacts, including artifacts where for other grey levels the vertical separation between pixels becomes too small.

Experiment 3

In this case the next scanline error impulse response function is a sampling of the Cauchy distribution (also known as the Lorentz distribution).

$$\left.\begin{aligned} h_{curr}[i] &= a \,.\, 1/(b^2 + i^2) \text{ for } i >= 1 \\ h_{next}[i] &= c \,.\, 1/(b^2 + i^2) \end{aligned}\right\} \quad (58)$$

where a, b, c are positive constants, with c chosen so that $\Sigma_i\, h_{next}[i] = 1$

The parameter b controls the spread of the distribution.

Observations for experiment 3

When a = 1 and b = 1, the halftone patterns for all grey levels with sparse halftone patterns are well spread. Also the general quality of the halftone output is high across all grey levels.

With 8 bit monochrome halftoning, the extreme sparse halftone patterns correspond to grey levels 1 and 254. By allowing fractional grey levels less than 1 the

behaviour of error diffusion can be observed for halftone patterns which are much more sparse than the halftone patterns for grey level 1.

It is unexpectedly noted that the very sparse halftone patterns for sub-unity fractional grey values are also well spread when the Cauchy distribution is used for the current and next scanline response functions. That is, error diffusion processing using the Cauchy distribution appears capable of generating well spread halftone patterns no matter how sparse the patterns.

When b is increased well above 1 the vertical separation of minority pixels in sparse halftone patterns is too small; when b is decreased well below 1 the horizontal separation of minority pixels in sparse halftone patterns is too small.

Experiment 4

In this case the next scanline error impulse response function is a generalisation of the Cauchy or Lorentz distribution.

$$\left.\begin{aligned} h_{curr}[i] &= a \,.\, 1/(b^2 + i^2)^p \text{ for } i >= 1 \\ h_{next}[i] &= c \,.\, 1/(b^2 + i^2)^p \end{aligned}\right\} \qquad (59)$$

where a, b, c, p are positive constants, with c chosen so that $\Sigma_i\, h_{next}[i] = 1$

Observations for experiment 4

With p set to be greater than 1, the observations are similar in character to those for experiments 1 and 2.

With p set to be less than 1 (but necessarily greater than 1/2), the halftone patterns for low or very high grey values are well spread horizontally, but the horizontal separation for intermediate grey values is too small.

Conclusion from the experiments

As noted, the Cauchy distribution of experiment 3 generates well spread halftone patterns across the entire range of grey values with sparse patterns, while the other

distributions fail to do so. It is believed the reason for the constrasting behaviours lies in the 'spread' of the distributions and in the spread of the self-convolutions of the distributions.

Expressing neighbourhood error in terms of the current and next scanline error impulse response functions and pixel only errors, from (2),(4), and (7) it is seen that:

$$e\_nbr_j = e\_curr\_line_j + e\_prev\_line_j \quad (59)^*$$

$$= (e_j + e\_prev\_line_j) * h_{curr} + e\_prev\_line_j \quad (60)$$

$$= (e_j + \Sigma_{l<j}\, e_l * h_{next}^{*(j-l)}) * h_{curr} + \Sigma_{l<j}\, e_l * h_{next}^{*(j-l)} \quad (61)$$

Accordingly, if the strength of the next scanline error impulse response function and its self convolutions is weak beyond some width, then insufficient account is taken of pixel only error at pixels horizontally separated by that width or more. Hence the disincentive to place minority pixel results at that width is insufficient, leading to sparse patterns which are not spread enough horizontally.

Self-convolution of the Cauchy distribution

The Cauchy distribution is given by

$$f(x) = (1/\pi)\,(b / (b^2 + x^2)) \quad (17)$$

where x is a real number and b is a real positive constant.

By change of variable from x to θ, where tan θ = b / x, it can be shown that,

$$\int_{-\infty}^{\infty} \left(\frac{1}{\pi}\right) \frac{b}{(b^2 + x^2)}\, dx = 1 \quad (62)$$

The Cauchy distribution satisfies the following self-convolution equations:

$$(1/\pi)\,(b / (b^2 + x^2)) * (1/\pi)\,(b / (b^2 + x^2)) = (1/\pi)\,(2b / ((2b)^2 + x^2)) \quad (63)$$

and for n = 1, 2, 3 ..

$$(1/\pi)\,(b / (b^2 + x^2))\ *^n = (1/\pi)\,(nb / ((nb)^2 + x^2)) \quad (18)$$

These results can be established from Fourier Transform theory. The following definitions and notation for the Continuous Space Fourier Transform are used:

Analysis equation / forward transform: $F(w) = \int_{-\infty}^{\infty} f(x)\, e^{-jwx}\, dx$ (64)

Synthesis equation / reverse transform $f(x) = (1/2\pi) \int_{-\infty}^{\infty} F(w)\, e^{jwx}\, dw$ (65)

Fourier Transform pair: $f(x) \Leftrightarrow F(w)$ (66)

The Fourier Transform pair corresponding to the double exponential distribution, with 'a' being a positive real constant, can be shown to be

$$e^{-a|x|} \Leftrightarrow 2a / (a^2 + w^2) \quad (67)$$

By the symmetry / duality property of the Fourier Transform

$$(1/\pi)\, a / (a^2 + x^2) \Leftrightarrow e^{-a|w|} \quad (68)$$

By the convolution property of the Fourier Transform, with b = a,

$$((1/\pi)\, b / (b^2 + x^2))\ *^n \Leftrightarrow e^{-nb|w|} \quad (69)$$

From this, (18) can be deduced. It can be stated that the Cauchy distribution is determined by its Continuous Space Fourier Transform which is a two-sided exponential function.

Discrete convolution

Related equations exist for discrete convolution.

For $k \in \mathbf{Z}$, let

$$h[k] = (b / \pi)\, ((1 + (-1)^{k-1} e^{-b\pi}) / (b^2 + k^2)) \quad (19)$$

then for n = 1, 2, 3 ..

$$h[k]\ *^n = (nb / \pi)\, ((1 + (-1)^{k-1} e^{-nb\pi}) / ((nb)^2 + k^2)) \quad (20)$$

and

$$\Sigma_{k \in Z}\, h[k] = 1 \quad (21)$$

The above equations can be established using Fourier Transform theory, either using Fourier Series or using the Discrete Space Fourier Transform. The following notation is used for the Discrete Space Fourier Transform:

Synthesis equation: $h[k] = (1/2\pi) \int_{2\pi} H(e^{jw})\, e^{jwk}\, dw$ (70)

Analysis equation: $H(e^{jw}) = \Sigma_{k \in Z}\, h[k]\, e^{-jwk}$ (71)

the use of the argument, e^{jw}, rather than w, indicates that the function, H, is periodic in w, with period 2π, so that:

$$H(e^{j(w+2\pi)}) = H(e^{jw}) \qquad \text{for all w.} \tag{72}$$

The sequence (or discrete signal or discrete space function), h[k] of (19), has the Discrete Space Fourier Transform given by:

$$H(e^{jw}) = e^{-b|w|} \qquad \text{for } -\pi \le w \le \pi\,. \tag{73}$$

It can be stated that the discrete space function of (19) is determined by its Discrete Space Fourier Transform which is a replicated two-sided exponential function.

Equation (20) follows by considering the self-convolution of the function of equation 19 and from the convolution property for the Discrete Space Fourier Transform. (That is, the convolution of 2 discrete time functions is the product of their Fourier Transforms.)

Equation (21) follows by considering the value of H at w = 0.

The sequence, or discrete space function,

$$h[k] = (\sinh(b\pi)\, b / \pi)\, (-1)^k / (b^2 + k^2) \tag{74}$$

has the Discrete Space Fourier Transform given by

$$H(e^{jw}) = \cosh(bw) = (e^{bw} + e^{-bw}) / 2 \qquad \text{for } -\pi \le w \le \pi \tag{75}$$

Considering the value of H at $w = \pi$ gives

$$\Sigma_{k \in Z}\, (b / \pi)\, (1 / (b^2 + k^2)) = \coth(b\pi) \tag{22}$$

Considering equations (19) and (22) with b = 1, it is seen that $e^{-\pi}$ is much less than 1, being approximately 0.0432 and $\tanh(\pi) = 1 / \coth(\pi)$ is close to 1, being approximately 0.9963.

So, defining h[k] according to the following equation:

$$h[k] = (\tanh(\pi) / \pi) (1 / (1 + k^2)) \quad (23)$$

it is deduced by setting b=1 in (22) that:

$$\Sigma_{k \in Z} h[k] = 1 \quad (76)$$

Further, h[k] according to (23) satisfies the following approximation:

$$h[k] \approx (1 / \pi) (1 / (1 + k^2)) \quad (77)$$

and

$$h[k]^{*n} \approx \{(1 / \pi) (1 + (-1)^{k-1} e^{-\pi}) / (1 + k^2)\}^{*n} = (n / \pi) (1 + (-1)^{k-1} e^{-n\pi}) / (n^2 + k^2) \quad (78)$$

$$\approx (n / \pi) (1 / (n^2 + k^2)) \quad \text{for } n = 1, 2, 3 \ldots . \quad (79)$$

The function of equation (23) can be considered a discrete approximation to the (continuous) Cauchy distribution with 'b' parameter equal to 1; it is derived from that Cauchy distribution by sampling and by normalising so that the sum of the function values is 1.

<u>The Cauchy distribution spreads in proportion to the degree of self-convolution</u>

Considering (18), it is seen that the Cauchy distribution has the remarkable property that repeated self-convolution preserves the form of the distribution. The self convolution of degree n of the Cauchy distribution is a scaled copy of the original distribution on a scaled axis:

$$f(x)^{*n} = (1/\pi) (nb / ((nb)^2 + x^2)) = (1/n) (1/\pi) (b / (b^2 + (x/n)^2)) \quad (80)$$

So that

$$f(x)^{*n} = (1/n) f(x/n) \quad (24)$$

The variance of the Cauchy distribution is not finite. An alternative measure of the width of the distribution is provided by the "equivalent width" as described in "The Fourier Transform and its Applications" by R.N. Bracewell (page 148) as follows.

The equivalent width of a function is the width of the rectangle whose height is equal to the central ordinate and whose area is the same as that of the function:

$$\frac{\int_{-\infty}^{\infty} f(x)dx}{f(0)} \tag{81}$$

From (17) and (18), or from (24) it can be seen that the central ordinate of the self-convolution of degree n of the Cauchy distribution is 1/n times the central ordinate, f(0), of the Cauchy distribution. The area under f(x) is 1; and as a consequence the area under $f(x)^{*n}$ is also 1. So the equivalent width of the Cauchy distribution increases in proportion to the degree of self-convolution.

The flux density of a point source, conservative vector field through a straight line is described by the Cauchy distribution

The manner in which the Cauchy distribution spreads in proportion to the degree of self-convolution can also be shown graphically.

Fig. 12 shows a vector field, ***A***, with vector magnitude given by 1/r where r is the distance from a source point, ***O*** (ie. 1216), and vector direction pointing away from ***O***. This vector field is conservative in that the flux out of any simple closed curve, not enclosing the source point, is zero. The magnitude of the vector field can be considered as a signal which spreads uniformly in 2 dimensions out from ***O***, preserving its strength - that is, reducing in proportion to the length of its circular wavefront.

Considering the flux across a horizontal line, ***L*** (ie. 1202), which has ***P*** (ie. 1214), its nearest point to ***O*** (ie. 1216), at a distance b (ie. 1210) from ***O*** (ie. 1216), the flux density function, f(x), is defined as a function of the offset, x, along the line from ***P***, such that

$$\int_s^t f(x)dx \tag{82}$$

is the flux of the vector field through a segment [s,t] of line L.

Let the angle which a point at offset x along the line L, makes with the line from ***O*** to ***P*** be θ (ie. 1220), then f(x) is given by

$$f(x) = c\,(1/r)\cos\theta \tag{83}$$

with *c* being a constant.

Choosing *c* to be $1/\pi$ so that the total flux through the line L is 1, we have

$$f(x) = (1/\pi)\,(1/r)\,(b/r) = (1/\pi)\,b\,/\,(b^2 + x^2), \tag{84}$$

which is the Cauchy distribution.

The offset, x, and angle, θ, are related by:

$$x = b\tan\theta \tag{25}$$

$$dx/d\theta = b\,(1 + \tan^2\theta) \tag{85}$$

$$d\theta/dx = b\,/\,(b^2 + x^2) \tag{26}$$

Using (25) and (26), confirms that the flux through a segment [s,t] (ie. 1218) is proportional to the angle, θ_{st} (ie. 1224), subtended at ***O*** as shown in the following:

$$\int_{b\tan\theta}^{b\tan(\theta+\theta_{st})}\left(\frac{1}{\pi}\right)\frac{b}{b^2+x^2}dx = \frac{1}{\pi}\int_{\theta}^{(\theta+\theta_{st})}d\theta = \frac{\theta_{st}}{\pi} \tag{86}$$

Considering another line, L' (ie. 1204), with nearest point at a distance nb (ie. 1206) from ***O*** and which is parallel to L. Segment [ns, nt] (ie. 1222) is the projection of segment [s,t] (ie. 1218) onto the line L' (ie. 1204), and subtends the same angle θ_{st} (ie. 1224) at ***O*** (ie. 1216).

The flux through segment [ns, nt] (ie. 1222) is also equal to θ_{st}/π and is associated with a flux density function for line L' equal to the self-convolution of f(x) of degree n as follows:

$$\int_{ns}^{nt} f(x)^{*n} dx = \int_{ns}^{nt} \frac{1}{n} f(\frac{x}{n}) dx = \int_{ns}^{nt} \frac{1}{\pi} \frac{nb}{((nb)^2 + x^2)} dx = \int_{s}^{t} \frac{1}{\pi} f(u) du = \frac{\theta_{st}}{\pi} \quad (87)$$

So It is seen that flux density functions given by the Cauchy distribution and its self convolutions correspond to a signal radiating from a point source which preserves its strength within any wedge formed by 2 rays emanating from the signal source point.

Desirability of the next scanline error impulse response function spreading in proportion to the degree of self-convolution and approximating a Cauchy distribution

It has been shown that the spread of the next scanline error impulse response function and its self convolutions relates to the horizontal separation between minority pixels in sparse halftone patterns.

It has been shown by experiment that when the next scanline error impulse response function samples a Cauchy distribution with the a and b parameters close to 1, sparse halftone patterns are well spread and worm-free. As well, when the next scanline error impulse response function is derived from equation 19, sparse halftone patterns are again well spread and worm-free.

It has also been shown that self convolutions of the Cauchy distribution spread in proportion to the degree of self-convolution and maintain a radially uniform distribution of signal strength - neither shifting signal strength to the centre nor shifting it towards the extremities.

A real, symmetric distribution which spreads in proportion to the degree of self-convolution must be a Cauchy distribution

It can be shown that a real, symmetric continuous impulse response function which preserves the magnitude of the impulse, spreads in proportion to the degree of self-convolution and which acts as a smooth (continuous) low pass filter, dampening all non-zero frequencies must be a Cauchy distribution.

Consider a positive real valued impulse response function f(x).

The condition that f(x) spreads in proportion to the degree of self-convolution can be written as follows:

$$f(x) *^n = a\ f(x/n) \text{ for some constant } a \tag{88}$$

Let F(w) be the Fourier Transform of f(x). The condition that f(x) preserves the magnitude of the impulse can be written as follows:

$$F(0) = \int_{-\infty}^{\infty} f(x)\, dx = 1 \tag{89}$$

By reference to the Fourier convolution theorem, the condition that the magnitude of the impulse is preserved by self-convolution can be written as follows:

$$\int_{-\infty}^{\infty} f(x)*^n dx = [F(0)]^n = 1 \tag{90}$$

Also, the integral of f(x/n) over all real values is n. So it can be deduced that

$$a = 1/n \tag{91}$$

So f(x) satisfies (24) as follows:

$$f(x) *^n = (1/n)\ f(x/n) \tag{92}$$

Taking the Fourier Transform of each side of (24) gives the following:

$$(F(w))^n = F(nw) \text{ for } n = 1, 2, 3 .. \tag{93}$$

Differentiating the above equation with respect to w gives the following:

$$n\ .\ (F(w))^{n-1}\ .\ F'(w) = n\ .\ F'(nw) \tag{94}$$

By dividing (93) by (94), the following results:

$$F(w) / F'(w) = F(nw) / F'(nw) \text{ for } n = 1, 2, 3 .. \tag{95}$$

By the condition that F is continuous (smoothly dampening all non-zero frequencies) it can be deduced that:

$$\text{for } w >= 0,\ F(nw) / F'(nw) = c_1, \text{ a constant} \tag{96}$$

Solving this differential equation gives the following:

$$F(w) = c_2\ e^{B.w} \text{ for } w > 0 \text{ for constants } c_2 \text{ and } B \tag{97}$$

Because f(x) is real-valued and symmetric, F(w) must be also; so that c_2 and B are both real. For F(0) = 1, $c_2 = 1$. For f(x) to act as a low pass filter, c_3 must be negative. From the above, it can be deduced that:

$$F(w) = e^{-b|w|} \text{ for a positive constant b.} \quad (98)$$

Using the inverse Fourier Transform gives the following:

$$f(x) = (1/\pi)\, b / (b^2+x^2) \text{ as claimed.} \quad (99)$$

Cauchy error diffusion

The (continuous) Cauchy distribution spreads in proportion to the degree of self-convolution. It is believed that this spreading property is desirable for the (discrete) next scanline error impulse response function. Moreover, it is envisaged that this spreading property is desirable for error diffusion irrespective of the number of scanlines with mask positions, and is desirable for other neighbourhood halftoning techniques. Similarly, it is believed that it is desirable that, in some sense, the (discrete) next scanline error impulse response function should approximate a (continuous) Cauchy distribution.

The name "Cauchy error diffusion" is defined to be error diffusion where the next scanline error impulse response function is designed so that it spreads approximately in proportion to the degree of self convolution or where the next scanline error impulse response function is designed to approximate a Cauchy distribution.

Diagram of repeated convolution of a next scanline error impulse response function approximating a Cauchy distribution

The next scanline error impulse response function describes the contribution of pixel-only errors of pixels of a scanline to the neighbourhood errors of pixels of the next scanline just prior to processing the next scanline.

Similarly, the n^{th} next scanline error impulse response function can be defined as the contribution of pixel-only errors of pixels of a scanline to the neighbourhood errors of pixels of the n^{th} next scanline just prior to processing the n^{th} next scanline.

Fig. 13 shows the n^{th} next scanline error impulse response function where the next scanline error impulse response function approximates a Cauchy distribution. It shows the contribution of the pixel-only error of a reference pixel to the neighbourhood error of a pixel of interest, n scanlines below the reference pixel just prior to processing the scanline containing the pixel of interest. Fig. 13 shows a reference pixel 1308 and a pixel of interest 1316 separated from the reference pixel 1308 by a distance "k" depicted by an arrow 1310, in a horizontal direction, and by a distance "n" depicted by an arrow 1312 in a vertical direction.

It is noted that the next scanline 1318 is the scanline immediately following the scanline in which the reference pixel 1308 is located. The function 1302 is the approximate next scanline impulse response function which determines distribution of error from the reference pixel 1308 to the next scanline 1318. The function 1304 is the approximate second next scanline impulse response function representing distribution of error from the reference pixel 1308 to scanline 1320. The function 1306 is the approximate n^{th} next scanline impulse response function representing distribution of error from the reference pixel to scanline 1322. A bold arrow 1314 represents a radial separation between the reference pixel 1308 and the pixel of interest 1316, noting that the radial separation is provided by the relationship $r=(n^2+k^2)^{1/2}$, and the approximate impulse response is represented by $(1/\pi)n/r^2$.

In this diagram, and hereafter, the Cauchy distribution is referred to with a spread parameter, b, equal to 1. Spread parameters close to 1 produce well spread sparse

halftone patterns; whereas spread parameters further removed from 1 generate poor quality halftone patterns.

In the diagram, the identical fan of arrows on each scanline is intended to portray that the same next scanline impulse response function applies for the processing of each pixel of each scanline.

As has been seen, for the next scanline error impulse response function, h_{next}, given by (23):

$$h_{next}[k]^{*n} \approx (n / \pi)\ 1 / (n^2 + k^2) = (n / \pi)\ 1 / r^2 \qquad (100)$$

So, the cumulative effective of distributing error according to a next scanline function approximating $(1 / \pi)\ 1 / (1 + k^2)$ is that the impulse response of error at the reference pixel on the pixel of interest, prior to processing the scanline of the pixel of interest, is approximately $(n / \pi)\ (1 / (n^2 + k^2) = (n / \pi)\ 1 / r^2$. Here r is the distance between the reference pixel and the pixel of interest.

That is, the pixel-only error at a reference pixel contributes to the neighbourhood errors of pixels of any succeeding scanline, just prior to processing that scanline, approximately in mutual proportions of $1/r^2$. The factor (n / π) simply scales all the contributions of the pixels of the particular succeeding scanline.

It is interesting to note that the impulse response function, c / r^2, with c a constant, is the only radially symmetric function in 2 dimensions for which the impulse response on a region is independent of scale.

To see this, the impulse response on an infinitesimal region, dA, can be described as $(dA\ .\ c/r^2)$. With change of scale by a factor, α, so that horizontal and vertical separations x and y instead measure αx and αy, the area of the same infinitesimal region now measures $\alpha^2 dA$, and the impulse response on the infinitesimal region is given by:

$$\alpha^2 dA.\ c / (\alpha r)^2 = dA\ .\ c/r^2 \qquad (101)$$

which is unchanged.

Graphs of self convolutions of the next scanline error impulse response function

Figs. 14, 15, 16 show graphs of self convolutions of the next scanline error impulse response function for:

1. Floyd Steinberg error diffusion;
2. the error diffusion of U.S. Patent No. 5,353,127 (Shiau & Fan); and
3. where the function is a scaled sampling of a Cauchy distribution.

Each of these graphs are discussed in relation to the halftone patterns generated by corresponding error diffusion algorithms performing monochrome bi-level halftoning of 8 bit per pixel image data.

Fig. 14 shows a graph 1400 of self convolutions of the next scanline error impulse response function for Floyd Steinberg error diffusion. The graph in Fig. 14 is plotted in regard to an abscissa 1404 which is defined in terms of "horizontal pixel offset" from a pixel of interest, and an ordinate 1402 measured in terms of impulse response amplitude. A legend 1406 is provided in regard to "next scanline response" (ie. self-convolution of degree 1) 1408, "self convolution" (ie. self-convolution of degree 2) 1410, "self convolution order 3" 1412, "self convolution order 4" 1414, and "self convolution order 5" 1416. In Fig. 14, the self-convolutions are not left-right symmetric and spread less rapidly than in Fig. 16 where they spread in proportion to the degree of convolution. The sparse halftone patterns of Floyd Steinberg error diffusion exhibit severe worm artifacts and strong left-right asymmetry.

Fig. 15 shows a graph 500 of self convolutions of the next scanline error impulse response function for the patent of Shiau and Fan. The graph in Fig. 15is plotted in regard to the abscissa 504 measured in terms of "horizontal pixel offset" from a pixel of

interest, and an ordinate 502 measured in terms of impulse response amplitude. A legend 506 is provided in regard to "next scanline response" 508, "self convolution" 510, "self convolution order 3" 512, "self convolution order 4" 514, and "self convolution order 5" 516. In Fig. 15, the self-convolutions are nearly left-right symmetric, but spread less rapidly than in Fig. 16 where they spread in proportion to the degree of convolution. For the error diffusion of U.S. Patent No. 5,353,127, worm artifacts are only apparent for very sparse halftone patterns. As the halftone patterns become very sparse, the artifacts become severe. The worm artifacts show little left-right asymmetry.

Fig. 16 shows a graph 600 of self convolutions of the next scanline error impulse response function for error diffusion as described in relation to Fig. 13. The graph in Fig. 16 is plotted in regard to the abscissa 604 measured in terms of "horizontal pixel offset" from a pixel of interest, and an ordinate 602 measured in terms of impulse response amplitude. A legend 606 is provided in regard to "next scanline response" 608, "self convolution" 610, "self convolution order 3" 612, "self convolution order 4" 614, and "self convolution order 5" 616. In Fig. 16, the self-convolutions are left-right symmetric and spread in proportion to the degree of convolution. In particular, it can be seen that the central peak value decreases as 1/n. For error diffusion as described for experiment 3 where the next scanline error impulse response function, h_{next}, is a sampling of the Cauchy distribution $(1/\pi)\ (1 / (1 + x^2))$, normalised so that $\Sigma_i\ h_{next}[i] = 1$, and all scanlines are processed left to right, there are no worm artifacts. The minority pixels in all sparse halftone patterns are well spread. Also, the sparse halftone patterns show very little left-right asymmetry. The occasional slight left-right asymmetry can be removed by varying the scanline processing direction.

As previously stated, it is believed that it is desirable that the next scanline error impulse response function be left-right symmetric and that it should spread in proportion

to the degree of self-convolution. These graphs provide a picture of how existing error diffusion methods fall short in this regard.

A method for generating error diffusion masks for which the next scanline error impulse response function approximates a Cauchy distribution

In implementing error diffusion, it is desirable to minimise the processing per pixel.

As stated previously, the processing model of Fig. 11, where error diffusion processing per pixel is described in terms of the current and next scanline error impulse response functions, is useful for understanding desirable characteristics of error diffusion. However, for implementing error diffusion where the next scanline error impulse response function approximates a Cauchy distribution, the processing model of Fig. 11 is not very suitable.

This is because the Cauchy distribution reduces slowly away from the peak value, so that a large number of non-zero values of the next scanline error impulse response function are required to provide a good approximation, with consequently a large amount of processing per pixel.

By contrast, the processing models of Figs. 9 and 10 are better suited to implementing Cauchy error diffusion, because a desired next scanline error impulse response function can be well approximated using a comparatively small number of mask positions.

A method is now described for designing an error diffusion mask, for which the corresponding next scanline error impulse response function approximates a Cauchy distribution.

The difference is measured between the actual next scanline error impulse response function corresponding to an error diffusion mask and the desired target next

scanline error impulse response function using an objective function, Obj(), equal to the sum of the squares of the differences of the function values.

$$Obj(mask_{curr}[], mask_{next}[]) = \Sigma_k (h_{next\_actual}[k] - h_{next\_target}[k])^2 \quad (102)$$

The error diffusion mask weight values described here were generated for a target next scanline error impulse response function of the form of equation 23. (Alternatively, a target next scanline error impulse response function of the form of equation 19 is also appropriate). The actual objective function used to determine the error diffusion mask weight values is given by:

$$Obj(mask_{curr}[], mask_{next}[]) = \Sigma_{-N <= k <= M} (h_{next\_actual}[k] - (\tanh(\pi) / \pi) / (1 + k^2))^2 \quad (103)$$

where:

N is the number of next scanline mask positions to the left of the current pixel

M = 30.

It is desired to find $mask_{curr}[]$ and $mask_{next}[]$ such that

$$\left.\begin{array}{l} \Sigma_i\, mask_{curr}[i] + \Sigma_i\, mask_{next}[i] = 1 \\ Obj(mask_{curr}[], mask_{next}[]) \text{ is a minimum} \end{array}\right\} \quad (104)$$

It is noted from (12), that

$$\left.\begin{array}{l} h_{next} = (\delta + h_{curr}) * mask_{next} \\ (\delta + h_{curr}) \text{ is a casual function.} \end{array}\right\} \quad (105)$$

where

For h_{next} to be left-right symmetric as required, $mask_{next}$ should be an anti-causal function. That is, mask positions can be omitted on the next scanline ahead of the current pixel. It is only necessary to design an error diffusion mask coefficients for pixel offsets:

$$\left.\begin{array}{l} \text{with } i > 0 \text{ on the curent scanline } (mask_{curr}[]), \text{ and} \\ \text{with } \; i \le 0 \text{ on the next scanline } (mask_{next}[]) \end{array}\right\} \quad (106)$$

Avoiding use of mask coefficients for pixel offsets on the next scanline with $i > 0$, avoids error distribution processing associated with those pixel offsets, making for more efficient error diffusion implementations.

An approximate solution to this design problem can be solved by optimisation methods, including the method of steepest descent and the conjugate gradient method as documented, for example, in the textbook: "Applied Numerical Methods for Engineers" by R.J. Schilling & S.L. Harris. This textbook also includes C software for performing the optimisations.

The constraint that the sum of the mask coefficients sum to 1 may be effectively removed by treating one of the desired error diffusion coefficients as a dependent variable. As a first step, to obtaining an error diffusion mask, it is necessary to decide how many (or which) mask positions to use on the current and next scanline. That is, an error diffusion mask can be designed to meet a pre-determined choice of mask positions and consequently a pre-determined limit of processing per pixel.

The coefficients for the error diffusion masks of Figs. 4 and 19 were determined as described above and using the conjugate gradient method with the aid of software supplied with the above textbook.

Size of error diffusion masks - achieving goals of halftone image quality and implementation efficiency

It has been found that increasing the number of current and next scanline error diffusion mask positions, improves the quality of sparse halftone patterns. Although judgement of the quality of halftone patterns is somewhat subjective, it has been found that, in order to completely remove worm artifacts and poor spreading of sparse halftone patterns for the bi-level halftoning of 8 bit per pixel images, it is necessary to use a mask

with the width of the error diffusion mask of Fig. 4. The mask of Fig. 4 has 12 positions on the current scanline and 13 positions on the next scanline and is called a "12-13 mask".

The mask of Fig. 4 is used in the description of the error diffusion method of the first embodiment. Smaller masks, which are optimised so that the next scanline error impulse response function approximates a Cauchy distribution, may also be used - this may be desirable to provide a better trade-off between halftone image quality and implementation efficiency.

A family of error diffusion masks for which the next scanline error impulse response function approximates a Cauchy distribution

The most sparse halftone patterns occur for source image regions with grey value just above the minimum grey value and just below the maximum grey value. As the grey value moves closer to the middle grey value, the halftone patterns become less sparse. For most intermediate grey levels, there is no advantage in using error diffusion masks as wide as the 12-13 mask.

It is possible to perform step 4 of the error diffusion processing per pixel, the distribution of error, according to a mask dependent on the pixel grey value. This approach is used in the error diffusion method of the second embodiment.

It has been previously observed that the quality of error diffusion halftoning can be improved by selectively using a larger error diffusion mask for low and high grey value regions. For example, in the paper "Error Diffusion Algorithm with Reduced Artifacts", from the Proceedings of IS&T's 45th Annual Conference, May 10-15, 1992, New Jersey, the author, R. Eschbach, discusses use of a combination of 2 error diffusion masks.

In his paper, Eschbach, also observes that it is necessary that the 2 masks should have similar structure so that the parts of an image using both masks do not show crossover artifacts.

The requirement that the next scanline error impulse response function should approximate a particular Cauchy distribution, provides a design goal for generating a family of compatible error diffusion masks.

The second arrangement uses such a family of masks, with a range of widths, to achieve similar halftone quality to that achieved by the first arrangement but with significantly reduced average processing per pixel.

Description of the First Arrangement

Bi-level monochrome halftoning

Fig. 17 is a block diagram 1700, describing the processing performed per pixel according to the first arrangement. The processing determines a halftone output value for a pixel from the pixel input value and stored error values which are sums of error distributed to the pixel from previously processed pixels. An input image value $g_{i,j}$ for a pixel of interest is provided on a line 1702 to an addition process 1704, from which is output, on a line 1706, a modified input image value for the pixel (i,j), ie. $g\_mod_{i,j}$, this being input to an addition process 1708. Thereafter, a table index value $table\_index_{i,j}$ is determined, in respect to a table 1726, as the sum of the modified input image value $g\_mod_{i,j}$, and the least significant bits of the table index value of a previously processed pixel which is provided on a line 1718. In the case of left to right scanline processing, the table index value is provided by (108) (see below), and in the case of right to left scanline processing, the table index value is provided by (109) (see below). It is noted that the table index value $table\_index_{i,j}$ is computed as the sum of the modified input image value for the pixel $g\_mod_{i,j}$ on the line 1706 and the least significant bits of the table index

value of the previously processed pixel, this having been provided to a latch 1716 and then output on the line 1718. The upper bits of the table index value are used via line 1712 to index the table. The lower bits of the table index value are saved in a latch 1716 for use in determining the table index value of the next pixel.

Considering a row 1733 of the table 1726, where the row 1733 has been defined by determination of the upper bits of the table index value $table\_index_{i,j}$, then commencing at the left hand side of the table 1726, it is seen that respective values C1, C2, ..., C11, C12 are provided on lines 1729, 1731, ..., 1732, and 1734 respectively to addition processes 1774, 1766, ..., 1750, and also to a latch 1746, respectively. Again considering the row 1733 of the table 1726, it is seen that values associated with the right hand side of the table 1726, and in particular with columns n0, n1, n2, ..., n11 and n12 are provided on lines 1740, 1738, 1736, ..., 1744 and 1742 respectively to a latch 1784, adder processes 1790, 1794, ..., 1711, and 1719 respectively. It is noted that an error diffusion mask depiction is provided at the top of Fig. 17. The mask refers to a current pixel 720, having on its right hand side twelve successive current scanline pixel positions c1 to c12 (these being collectively designated as reference numeral 1724), and having, on a next line, a set of 13 next scanline pixel positions n0 to n12 (collectively designated as 1722).

A tandem arrangement of successive latches and adder processes 1746, 1750, 1754, ..., 1762, 1766, 1770, 1774 and 1776 provide, on a line 1778, a total error $e\_curr\_line_{i,j}$, of errors distributed directly to pixel (i,j) from pixels of the current scanline, which is fed to an adder process 1780. In a similar fashion, a tandem arrangement of respective latches and adder processes 1784, 1790, 1792, 1794, 1798 ..., 1707, 1711, 1717, 1719 provide, on lines 1723 or 1721, a total of error distributed directly from pixels of the previous scanline, ie. $e\_prev\_line_{i+12,\ j+1}$ (in the case of right to left scanline processing), or $e\_prev\_line_{i-12,\ j+1}$ (in the case of left to right scanline processing). The

values $e\_prev\_line_{i-12,\ j+1}$, or $e\_prev\_line_{i+12,\ j+1}$ are directed, by the lines 1723 or 1721 respectively, to an error line store memory 1725. The memory 1725 outputs, on a line 1727, the total error $e\_prev\_line_{i,j}$ distributed directly to the pixel (i,j) from pixels of the previous scanline on a line 1727, this being input to the adder process 1780. The adder process 1780 outputs, on a line 1782, the neighbourhood error value $e\_nbr_{i,j}$ at the pixel, this being the sum of errors distributed directly to the pixel, on a line 1782, which is input to the adder process 1704.

In this implementation, the halftone output value on the line 1730 and the errors distributed from a pixel are retrieved from a pre-calculated error distribution and result table 1726. The error values stored in the table have extra precision to store fractional error values. A suitable amount of storage per error value is 16 bits, with the most significant 8 bits representing the sign of the error together with the integer part of the error as a number in the range (-128,127], and with 8 least significant bits representing the fractional part of the error.

The same processing cycle is repetitively applied, pixel by pixel across a scanline, to process a scanline of pixels. The scanline processing is repeated for each scanline to halftone the image. The scanline processing direction for each scanline may either be left to right or right to left, and the scanline processing direction may change from one scanline to another. Some desirable halftone pattern randomisation is achieved by varying the scanline processing direction.

The halftone output value on the line 1730 for a pixel at position (i, j) (being column i and scanline j) is determined according to the pixel decision rule defined by the following equation.

$$\text{if } g_{i,j} + e\_nbr_{i,j} > th \text{ then } r_{i,j} = 255; \text{ else } r_{i,j} = 0 \qquad (27)$$

where:

$g_{i,j}$ is the input image value of the pixel;

$e\_nbr_{i,j}$ is the neighbourhood error value at the pixel, being the sum of errors distributed to the pixel;

th is a threshold value; and

$r_{i,j}$ is the halftone output value of the pixel (0 or 255).

It is convenient, though not necessary for the threshold value to be 127, as this then constrains the total error distributed from a pixel to the range (-128,127] and each error value stored in the table can be scaled up by 256 and represented as a signed 'short' integer in the range (-128*256, 127*256]. With error scaled up by 256 it is also necessary to scale up input values by 256. The table index value is then constrained to the range (-128,127] x 256 and can be stored in 17 bits. Choosing the upper 9 bits of the table index value to index the table produces good halftone output. It is also possible to increase the number of bits of the table index used to index the table, in order to allow avoiding the use of the lower bits of the table index value altogether; however, this also means a consequent increase in the size of the table.

Determination of a pixel's halftone output value according to the pixel decision rule, as well as the determination of the errors to distribute, is achieved by a table lookup. The table lookup is made using the most significant bits of a table index value computed as follows.

Firstly, the modified input image value for the pixel (i,j), $g\_mod_{i,j}$, is computed as follows:

$$g\_mod_{i,j} = g_{i,j} + e\_nbr_{i,j} \tag{107}$$

Secondly, the table index value, $table\_index_{i,j}$, is computed as the sum of the modified input image value for the pixel and the least significant bits of the table index

value of the previously processed pixel, this being dependent upon the scan direction as follows:

left to right scanline processing

$$table\_index_{i,j} = g\_mod_{i,j} + least\ significant\ bits(table\_index_{i-1,j}) \quad (108)$$

right to left scanline processing

$$table\_index_{i,j} = g\_mod_{i,j} + least\ significant\ bits(table\_index_{i+1,j}) \quad (109)$$

The error distribution and result table stores the halftone output value and error distribution values corresponding to each possible value of the most significant bits of a table index value.

The halftone output value is determined by a thresholding equation similar to (27) as follows:

$$if\ table\_index\_upper\_bits > th\ then\ out = 255;\ else\ out = 0 \quad (110)$$

The error distribution amounts are initially determined as

$$\left.\begin{aligned} e\_c[i] &= (table\_index\_upper\_bits - out) * mask_{curr}[i]\ for\ 1 <= i <= 12 \\ e\_n[i] &= (table\_index\_upper\_bits - out) * mask_{next}[i]\ for\ 0 <= i <= 12 \end{aligned}\right\} \quad (111)$$

The rounding errors in representing these error distribution amounts as 16 bit values are then distributed amongst the error distribution amounts so that

$$\Sigma_{1<=i<=12}\ e\_c[i] + \Sigma_{0<=i<=12}\ e\_n[i] = table\_index\_upper\_bits - out \quad (112)$$

The error resulting by thresholding a pixel is distributed by retrieving error distribution amounts from the error distribution and result table. These values are stored and summed with previous error distribution amounts and sums of error distribution amounts to deliver the error sum values given by formulae, as a function of scan direction as follows:

left to right scanline processing

$$\left.\begin{aligned} e\_curr\_line_{i+1,j} &= e\_c[1]_{i,j} + e\_c[2]_{i-1,j} + .. + e\_c[12]_{i-11,j} \\ e\_prev\_line_{i-12,j+1} &= e\_n[12]_{i,j} + e\_n[11]_{i-1,j} + .. + e\_n[0]_{i-12,j} \end{aligned}\right\} \quad (113)$$

right to left scanline processing

$$\left.\begin{aligned} e\_curr\_line_{i-1,j} &= e\_c[1]_{i,j} + e\_c[2]_{i+1,j} + .. + e\_c[12]_{i+11,j} \\ e\_prev\_line_{i+12,j+1} &= e\_n[12]_{i,j} + e\_n[11]_{i+1,j} + .. + e\_n[0]_{i+12,j} \end{aligned}\right\} \quad (114)$$

where $e\_curr\_line_{i,j}$ is the total error distributed directly to pixel (i,j) from pixels of the current scanline, and $e\_prev\_line_{i,j}$ is the total error distributed directly to pixel (i,j) from pixels of the previous scanline.

The error value, $e\_prev\_line_{i-12,j+1}$ (left to right processing) or $e\_prev\_line_{i+12,j+1}$ (right to left processing), is written to line store memory from which it will be retrieved as part of processing the next scanline.

The neighbourhood error value for the pixel (i,j), $e\_nbr_{i,j}$, is derived as the sum of the error sum value, $e\_curr\_line_{i,j}$, generated as part of processing the previous pixel of the current scanline, and the error sum value, $e\_prev\_line_{i,j}$, retrieved from the line store memory.

Description of a Second Arrangement

Bi-level monochrome halftoning

The second arrangement is also described for the case of halftoning a monochrome image.

This second arrangement uses a family of error diffusion masks, all specially designed so that for each mask, the next scanline error impulse response function, corresponding to the mask, approximates the same target function; that target function being a Cauchy distribution which is sampled and normalised. By these means, the generated sparse halftone patterns are free of worm artifacts but at the same time the processing involved in the first arrangement is reduced.

Fig. 19 shows an example of a suitable family 1900 of error diffusion masks for the second arrangement. The family 1900 of error diffusion masks comprises a first column 1902 defining a pixel offset in terms of the variables c1 to c12, and n0 to n12.

Error diffusion masks, depicted by columns 1904 to 1926, provide numerical values associated with the aforementioned pixel offsets, the masks being designated as mask 1 to mask 12 respectively.

The pixel offsets in the left column of the table correspond to the mask positions shown in Fig. 3. Mask 1 of this family has weights for 1 pixel position on the current scanline and 2 pixel positions on the next scanline; mask 2 has weights for 2 pixel positions on the current scanline and 3 pixel positions on the next scanline; and so on. That is, as the mask number increases, the width of the mask increases. Mask 12 is the same mask as that described in Figs. 3 and 4. For each mask of Fig. 19, the current line mask positions are those closest to the current pixel, and the next line mask positions are those closest to the current pixel which are below or behind the current pixel.

Each mask of the family was prepared as described above, so that its next scanline error impulse response function approximates a target next scanline error impulse response function given by:

$$h_{next\_target}[k] = (\tanh(\pi) / \pi) / (1 + k^2)) \quad (115)$$

The target next scanline error impulse response function is derived from the Cauchy distribution $(1 / \pi)\ 1 / (1 + x^2)$ by sampling and normalisation.

The halftone output value for each pixel is obtained by error diffusion, with the error diffusion mask used to determine error distribution amounts for the pixel being selected according to the pixel's grey level.

Fig. 20 shows a table 2000 with a suitable mapping between grey levels and the masks of Fig. 19. The table 2000 comprises two columns 2002 and 2004, the first being a mask index running from mask no. 1 to mask no. 12, and the second column 2004 showing correspondence between grey level values and the various mask indices. Thus,

for example, mask index no. 6, designated by a reference numeral 2006, is suitable for use with grey levels 16 to 18, 237 to 239, 121, and 134.

For the masks of Fig. 19 and the grey level to mask mapping provided by the table of fig. 20, generated halftone patterns do not show crossover artifacts. That is, the transition between use of different masks is not apparent in the halftone output. All generated sparse halftone patterns are well spread and worm-free; and the appearance of the halftone output is very similar to that of the first embodiment. At the same time, the average number of addition operations involved in the second embodiment is significantly less than that in the first embodiment.

Fig. 18 shows a block diagram for the processing performed per pixel for the error diffusion algorithm of the second arrangement. An input image value $g_{i,j}$ for the pixel being considered is provided on a line 1802, this being directed as depicted by lines 1806 and 1804 respectively to an adder process 1824 and a mask index table 1808. The mask index table 1808 provides, in the first instance, a mask index value which is used, as depicted by the arrow 1810 to select a particular error distribution table eg. 1814, 1816, and 1818, from a family 1820 of error distribution tables. In addition, the mask index table 1808 provides two other numbers, these being the number of mask positions on the current scanline, and the number of mask positions on the next scanline, for the particular mask corresponding to the mask index value.

The adder process 1824 provides, on a line 1826, a modified input image value $g\_mod_{i,j}$ for the pixel (i,j), directing this to an adder process 1828. The adder process 1828 provides, on a line 1830, and thereafter, a line 1832, a table lookup index $table\_index_{i,j}$ which is determined, as described in relation to Fig. 17, and in accordance with (108) and (109). The table index value $table\_index_{i,j}$ is computed by the adder process 1828 as a sum of the modified input image value $g\_mod_{i,j}$ on the line 1826 and

the least significant bits of the table index value for the previously processed pixel this being provided by a line 1834, a latch 1836, and a line 1838 which feeds into the adder process 1828. The most significant bits of the table index $table\_index_{i,j}$ on the line 1832 are used to retrieve a halftone output value for the pixel $r_{i,j}$ on a line 1822, as well as error distribution values on the lines 1840 and 1842. The number of current scanline mask positions (not shown) and the current scanline error distribution amounts on the line 1840 are used to update the error sum values in a current line error buffer 1846.

The number of next scanline mask positions (not shown) and the next scanline error distribution amounts on the line 1842 are used to update the error sum values in a line store error buffer 1848. The current line error buffer 1846 provides an error sum value $e\_curr\_line_{i,j}$ being the total error distributed directly to pixel (i,j) from pixels of the current scanline on a line 1862 which is directed to an adder process 1860. The line store error buffer 1848 provides an error sum value $e\_prev\_line_{i-12,j}$ (for left to right scanline processing) on a line segment 1852, or alternatively $e\_prev\_line_{i+12,j}$ (for right to left scanline processing) on a dashed line 1854, for storing in an error line store memory 1856. The memory 1856 provides, on a line 1858, a value $e\_prev\_line_{i,j}$, ie., the total error distributed directly to the pixel (i,j) from pixels of the previous scanline, which is directed to an adder process 1860. The adder process 1860 outputs, a value for the neighbourhood error for the pixel (i,j), ie., $e\_nbr_{i,j}$ on a line 1844, which is directed to the adder process 1824. The nature of the processing performed per pixel is similar to that of the first embodiment. The differences in processing between the 2 embodiments are now described.

In the second embodiment, the pixel grey level on the line 1802 is used to retrieve a mask index value and 2 other numbers from a mask index table 1808. The other 2 numbers are the number of mask positions on the current scanline and the number

of mask positions on the next scanline, for the mask corresponding to the mask index value. The mask index value is used to select an error distribution table from the family of error distribution tables 1820. The most significant bits of the table index generated for the pixel and the 2 numbers representing the number of mask positions, are used to retrieve the halftone output value for the pixel and the error distribution values from the selected error distribution table.

The number of current scanline mask positions and the current scanline error distribution amounts are used to update the error sum values in a current line error buffer 1846 and to provide the error sum value, $e\_curr\_line_{i+1,j}$ (left to right scanline processing) or $e\_curr\_line_{i-1,j}$ (right to left scanline processing) for the next pixel to be processed for the scanline. The number of addition operations required for this step is the number of current scanline mask positions for the selected mask. This is typically much less than the 11 addition operations required in the first arrangement.

The number of next scanline mask positions and the next scanline error distribution amounts are used to update the error sum values in a line store error buffer 1848 and to provide the error sum value, $e\_prev\_line_{i-12,j}$ (left to right scanline processing) or $e\_prev\_line_{i+12,j}$ (right to left scanline processing) for storing in the error line store memory 1856. The number of addition operations required for this step is the number of next scanline mask positions for the selected mask. This is typically much less than the 12 addition operations required in the first arrangement.

In summary, the advantage of the second arrangement over the first, is that the average number of addition operations required is dramatically reduced. When the halftoning method is implemented in software, where each addition operation must be performed sequentially, this can significantly improve the execution speed.

The described arrangements can be implemented for error diffusion where the output halftoned image has more than 2 levels and where the source image is a colour image.

The method of Cauchy error diffusion can be practiced using a general-purpose computer system 2100, such as that shown in Fig. 21 wherein the processes of Figs. 9-10, 17-18 may be implemented as software, such as an application program executing within the computer system 2100. In particular, the steps of method of Cauchy error diffusion are effected by instructions in the software that are carried out by the computer. The software may be divided into two separate parts; one part for carrying out the Cauchy error diffusion methods; and another part to manage the user interface between the latter and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer effects an advantageous apparatus for Cauchy error diffusion.

The computer system 2100 comprises a computer module 2101, input devices such as a keyboard 2102 and mouse 2103, output devices including a printer 2115 and a display device 2114. A Modulator-Demodulator (Modem) transceiver device 2116 is used by the computer module 2101 for communicating to and from a communications network 2120, for example connectable via a telephone line 2121 or other functional medium. The modem 2116 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 2101 typically includes at least one processor unit 2105, a memory unit 2106, for example formed from semiconductor random access memory

(RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 2107, and an I/O interface 2113 for the keyboard 2102 and mouse 2103 and optionally a joystick (not illustrated), and an interface 2108 for the modem 2116. A storage device 2109 is provided and typically includes a hard disk drive 2110 and a floppy disk drive 2111. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 2112 is typically provided as a non-volatile source of data. The components 2105 to 2113 of the computer module 2101, typically communicate via an interconnected bus 2104 and in a manner which results in a conventional mode of operation of the computer system 2100 known to those in the relevant art. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program is resident on the hard disk drive 2110 and read and controlled in its execution by the processor 2105. Intermediate storage of the program and any data fetched from the network 2120 may be accomplished using the semiconductor memory 2106, possibly in concert with the hard disk drive 2110. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 2112 or 2111, or alternatively may be read by the user from the network 2120 via the modem device 2116. Still further, the software can also be loaded into the computer system 2100 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 2101 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable media may alternately be used.

The method of Cauchy error diffusion may also be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of Cauchy error diffusion. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

Industrial Applicability

It is apparent from the above that the arrangements described are applicable to the image processing industries.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

AUSTRALIA ONLY

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

The claims defining the invention are as follows:

1. A method of halftoning an image, said method comprising steps of:

determining an output value of a current pixel on a current scanline using a sum of an input value for the current pixel and a neighbourhood error value at the current pixel;

determining an error at the current pixel as the difference between (i) the sum of the input value for the current pixel and the neighbourhood error value at the current pixel, and (ii) the output value of the current pixel; and

adding a proportion of the error at the current pixel to neighbourhood error values at as yet unprocessed pixels of a subsequent scanline in accordance with a next scanline error impulse response; wherein said next scanline error impulse response:

approximates a function which spreads with self-convolution in proportion to a degree of self-convolution.

2. A method of halftoning according to claim 1, wherein:

the next scanline error impulse response is a member of a plurality of next scanline error impulse responses, each of said plurality of next scanline error impulse responses approximating the function which spreads with self-convolution in proportion to the degree of self-convolution;

each said member of said plurality of next scanline error impulse responses is associated with a corresponding error diffusion mask; and

a corresponding size of each said error diffusion mask increases with increasing grey value of a region to which said mask is applied.

3. A method of halftoning according to claim 1, wherein:

the next scanline error impulse response is a sampling of one of a Cauchy distribution and a Lorentz distribution, said sampling being normalised so that a sum of next scanline error impulse response values is unity.

4. A method of halftoning according to claim 1, wherein:

the next scanline error impulse response is left-right symmetric.

5. A method of generating an error diffusion mask suitable for use with any of the aforementioned methods.

6. An error diffusion mask suitable for use with any of the aforementioned methods.

7. A method of halftoning an image, said image comprising a plurality of pixels each having an input value and an assignable output value that can take on one of at least two output values, where pixels are processed scanline by scanline and scanlines are processed one at a time from the top of the image to the bottom of the image, and where a scanline is processed pixel by pixel either from left to right or from right to left, and where the processing for each pixel comprises the steps of:

(a) determining the output value of a current pixel using a sum of the input value of the current pixel and a neighbourhood error value for the pixel;

(b) determining an error at the current pixel as the difference between, firstly, the sum of the input value of the current pixel and the neighbourhood error value for the pixel, and secondly the output value of the pixel;

(c) adding proportions of the error at the current pixel to the neighbourhood error values of yet to be processed pixels of the current and next scanline;

and where the said proportions of the error at a current pixel are designed so that the next scanline error impulse response, being that function which maps

(A) from a horizontal pixel offset;

(B) to the total proportion of the error at the current pixel added to the neighbourhood error of that pixel of the next scanline which is displaced by the horizontal pixel offset from the current pixel, following complete processing of the current scanline;

approximates a function which spreads with self-convolution in proportion to the degree of self-convolution.

8. A method as claimed in claim 7, where the next scanline error impulse response approximates a scaled sampling of a Cauchy distribution.

9. A method as claimed in claim 7, where the next scanline error impulse response approximates a function which has a Discrete Space Fourier Transform which is a replicated two-sided exponential function.

10. A method as claimed in claims 7 to 9, where the output value of a current pixel is determined by comparison of the sum of the input value of the current pixel and the neighbourhood error value for the pixel against a threshold value.

11. A method as claimed in claim 10, where in step (c), for a current pixel at pixel position (i,j), being column i and scanline j, error at the current pixel is added to the neighbourhood error of only those pixels which are either:

(i) on the current scanline ahead of the current pixel at a pixel position (i+current_offset, j), where, for left to right processing of the current scanline, current_offset is greater than zero, and, for right to left processing of the current scanline, current_offset is less than zero, or

(ii) on the next scanline below or behind the current pixel at a pixel position, (i-next_offset, j+1), where, for left to right processing of the current scanline, next_offset is greater than or equal to zero, and, for right to left processing of the current scanline, next_offset is less than or equal than zero.

12. A method of halftoning an image, said image comprising a plurality of pixels each having an input value and an assignable output value that can take on one of at least two output values, where pixels are processed scanline by scanline and scanlines are processed one at a time from the top of the image to the bottom of the image, and where a scanline is processed pixel by pixel either from left to right or from right to left, and where the processing for each pixel comprises the steps of:

(a) determining the output value of a current pixel using a sum of the input value of the current pixel and a neighbourhood error value for the pixel;

(b) determining an error at the current pixel as the difference between, firstly, the sum of the input value of the current pixel and the neighbourhood error value for the pixel, and secondly the output value of the pixel;

(c) selecting, using the current pixel input value, a set of proportions and a corresponding pixel position offsets, from a family of sets of proportions and corresponding pixel position offsets;

(d) adding the selected proportions of the error at the current pixel to the neighbourhood error values of yet to be processed pixels at pixel positions offset from the current pixel by the selected corresponding pixel position offsets;

and where each set of the said family of sets of proportions and corresponding pixel offsets, is designed so that the next scanline error impulse response corresponding to that set, being that function which maps

(A) from a horizontal pixel offset;

(B) to the proportion of the error at the current pixel added to the neighbourhood error of that pixel of the next scanline displaced by the horizontal pixel offset from the current pixel, following complete halftone processing of the current scanline using only the said set of proportions and corresponding pixel offsets;

approximates a function which spreads with self-convolution in proportion to the degree of self-convolution.

13. A method as claimed in claim 12, where each next scanline error impulse response, corresponding to a set of proportions and pixel offsets, approximates a scaled sampling of a Cauchy distribution.

14. A method as claimed in claim 12, where each next scanline error impulse response, corresponding to a set of proportions and pixel offsets, approximates a function which has a Discrete Space Fourier Transform which is a replicated two-sided exponential function.

15. A method as claimed in claim 13 or 14, where the family of sets of proportions and pixel offsets together with the selection, using the current pixel input value, of a set of

proportions and pixel offsets, are designed so as to minimise processing while also minimising the presence of artifacts in the halftone output artifacts including cross-over artifacts and poor spreading in sparse halftone patterns.

16. A method as claimed in claim 15, where the maximum absolute offset in each set of the family of sets of proportions and pixel offsets, varies so that the family of sets includes a set with small maximum absolute offset and a set with large maximum absolute offset, and where intermediate input values primarily select sets with small maximum absolute offset, and extreme input values primarily select sets with large maximum absolute offset.

17. A method as claimed in claim 16, where the output value of a current pixel is determined by comparison of the sum of the input value of the current pixel and the neighbourhood error value for the pixel against a threshold value.

18. A method as claimed in claim 17, where in step (d), for a current pixel at pixel position (i,j), being column i and scanline j, error at the current pixel is added to the neighbourhood error of only those pixels which are either:

(i) on the current scanline ahead of the current pixel at a pixel position (i+current_offset, j), where, for left to right processing of the current scanline, current_offset is greater than zero, and, for right to left processing of the current scanline, current_offset is less than zero, or

(ii) on the next scanline below or behind the current pixel at a pixel position, (i-next_offset, j+1), where, for left to right processing of the current scanline,

next_offset is greater than or equal to zero, and, for right to left processing of the current scanline, next_offset is less than or equal than zero.

19. A method of halftoning an image, said image comprising a plurality of pixels each having an input value and an assignable output value that can take on one of at least two output values, where pixels are processed scanline by scanline and scanlines are processed one at a time from the top of the image to the bottom of the image, and where a scanline is processed pixel by pixel either from left to right or from right to left, and where the processing for each pixel comprises the steps of:

(a) determining the output value of a current pixel using a sum of the input value of the current pixel and a neighbourhood error value for the pixel;

(b) determining an error at the current pixel as the difference between, firstly, the sum of the input value of the current pixel and the neighbourhood error value for the pixel, and secondly the output value of the pixel;

c) selecting, using the current pixel input value, a set of proportions and a corresponding pixel position offsets, from a family of sets of proportions and corresponding pixel position offsets;

(d) adding the selected proportions of the error at the current pixel to the neighbourhood error values of yet to be processed pixels at pixel positions offset from the current pixel by the selected corresponding pixel position offsets;

and where each set of the said family of sets of proportions and corresponding pixel offsets, only includes pixel offsets corresponding to pixels on the same scanline as the current pixel or to pixels on the next scanline.

Dated 29 December, 2000
Canon Kabushiki Kaisha
Patent Attorneys for the Applicant/Nominated Person
SPRUSON & FERGUSON

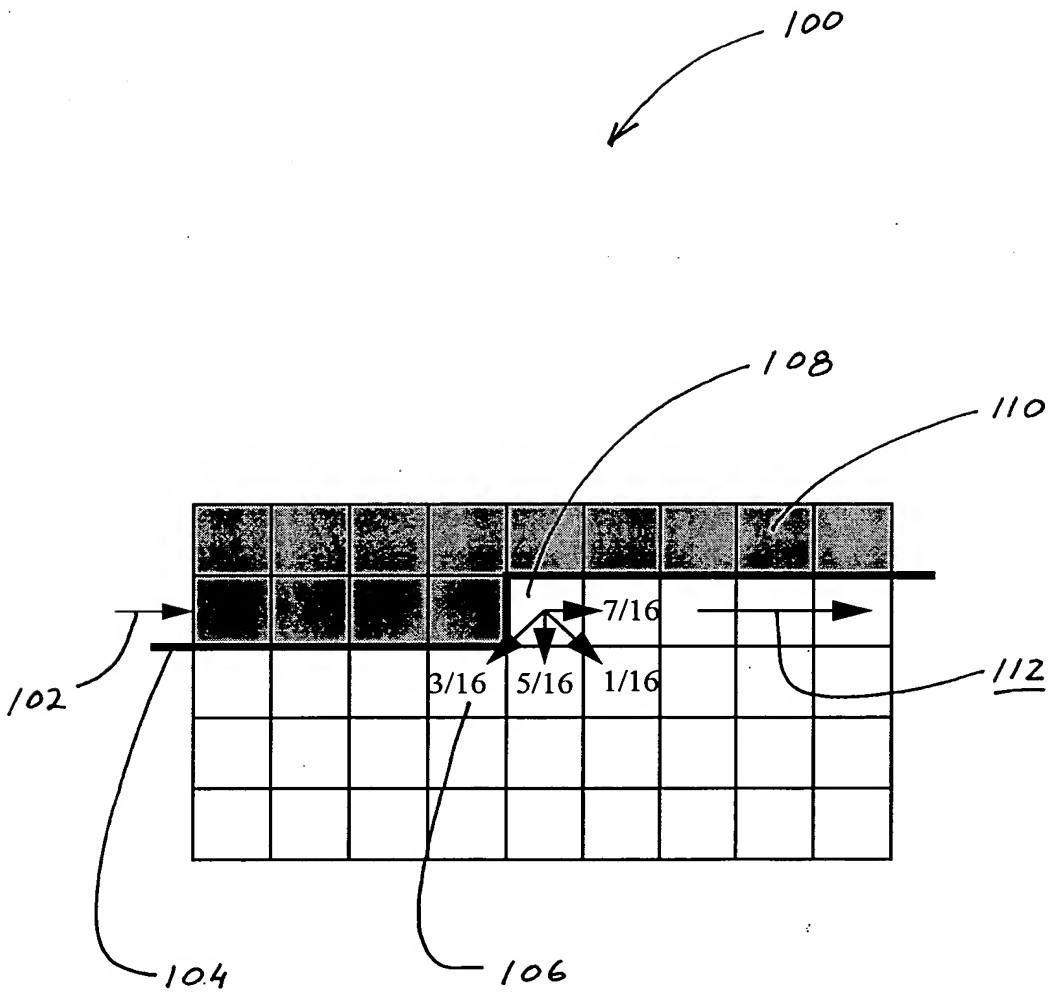

Fig. 1
(Prior Art)

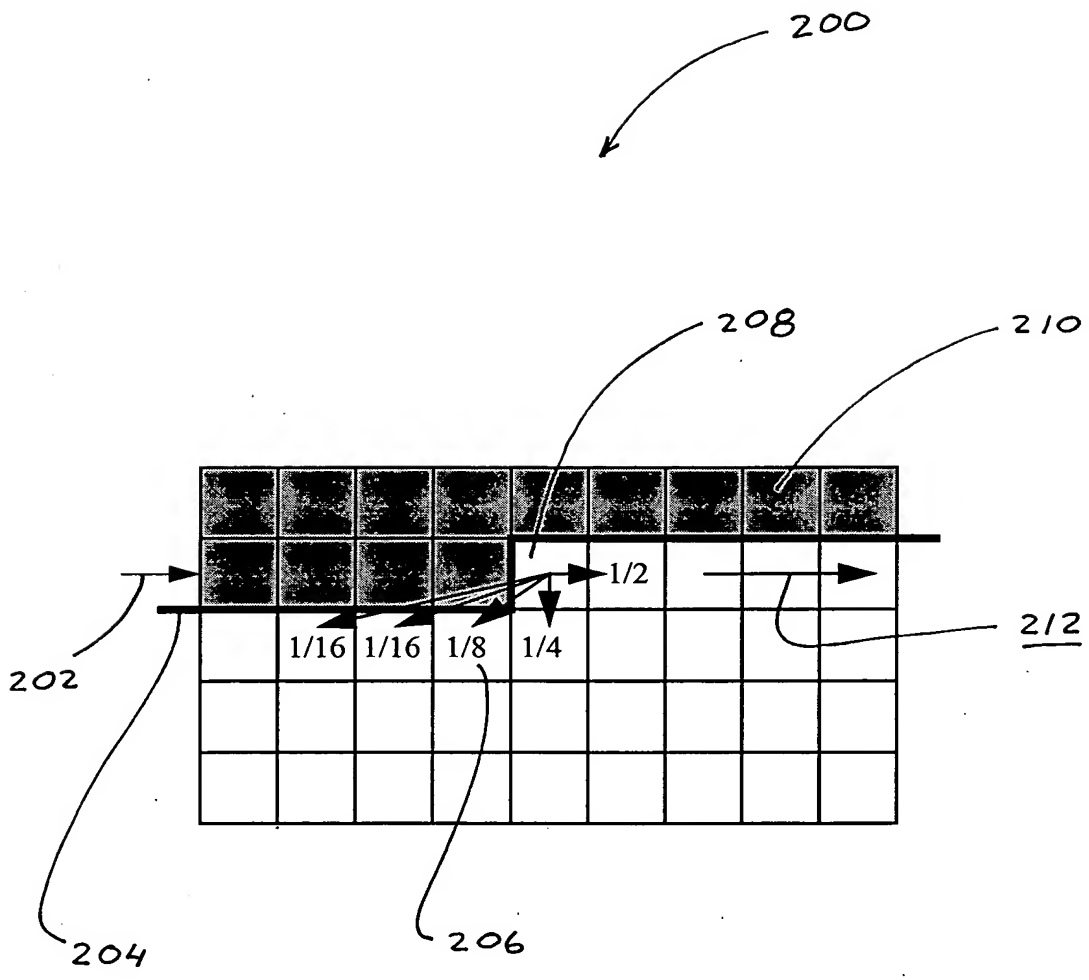

Fig. 2
(Prior Art)

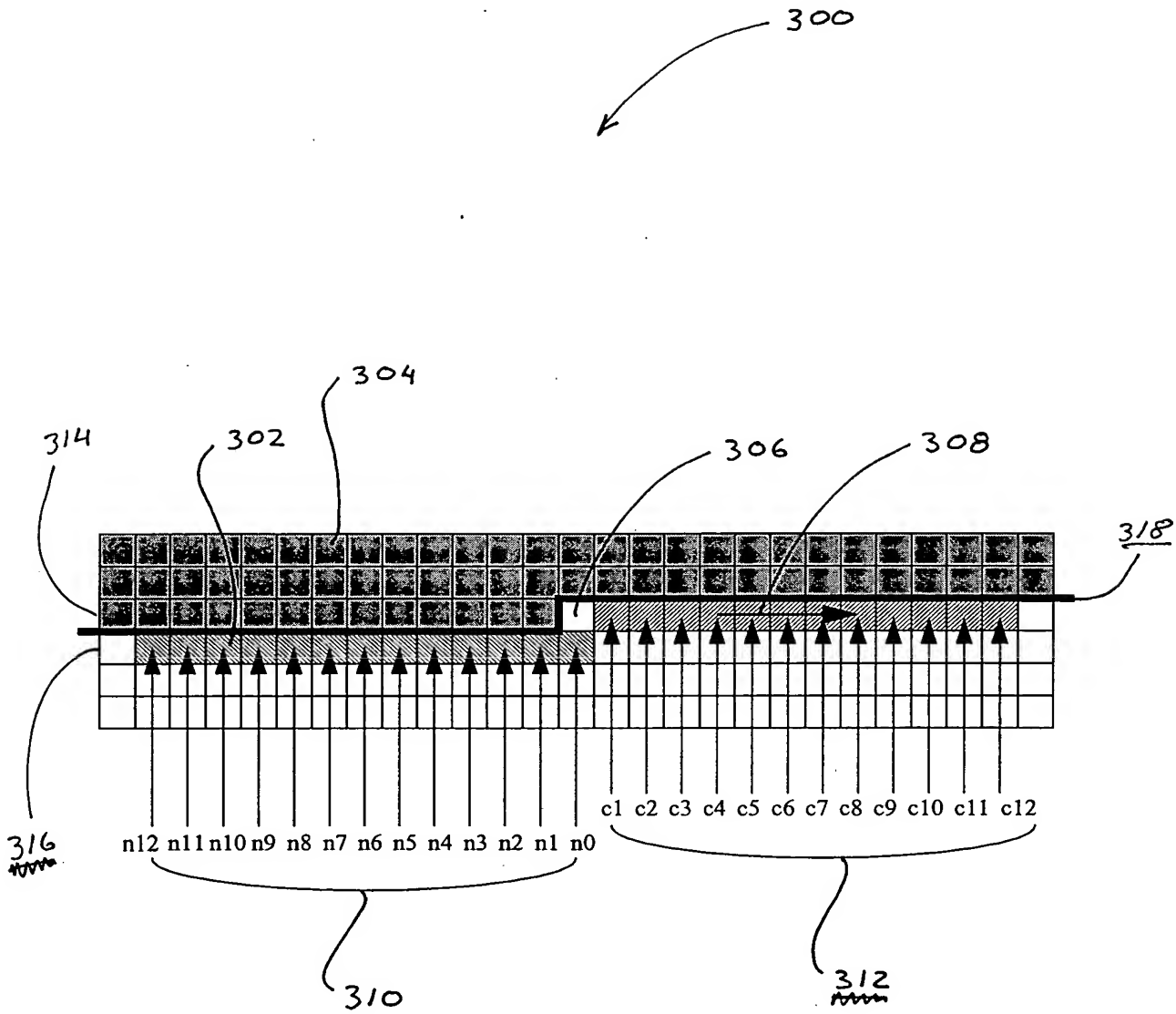
300
304
314
302
306
308
318
316
n12 n11 n10 n9 n8 n7 n6 n5 n4 n3 n2 n1 n0
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12
310
312

Fig. 3

400

mask position	mask weight	mask position	mask weight
n0	0.236962		
n1	0.127126	c1	0.534915
n2	0.048164	c2	-0.080892
n3	0.023072	c3	0.0286
n4	0.013238	c4	0.007557
n5	0.008605	c5	0.003258
n6	0.006122	c6	0.005601
n7	0.004631	c7	0.002344
n8	0.003692	c8	0.002784
n9	0.003043	c9	0.00211
n10	0.002717	c10	0.005269
n11	0.002068	c11	-0.005752
n12	0.003602	c12	0.011165

402 404 406 408

Fig. 4

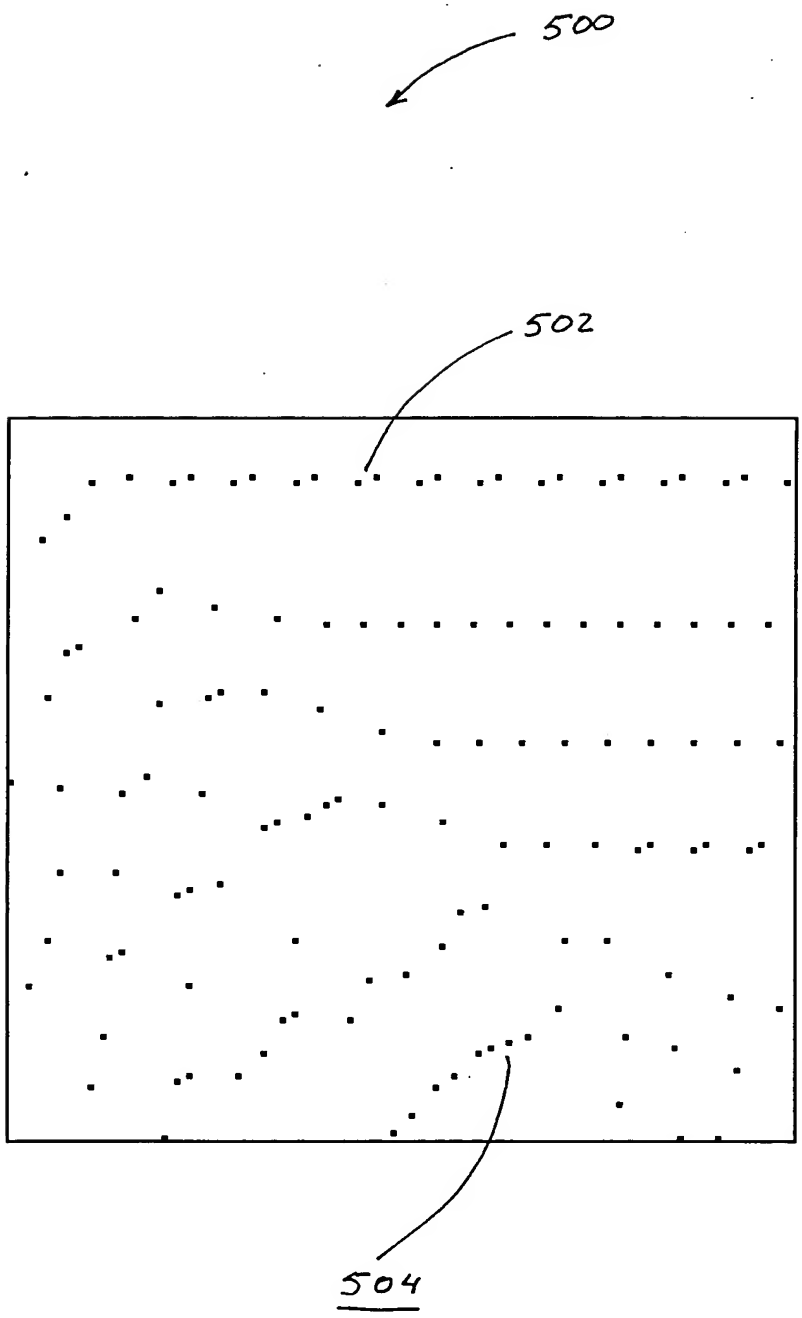

Fig. 5
(Prior Art)

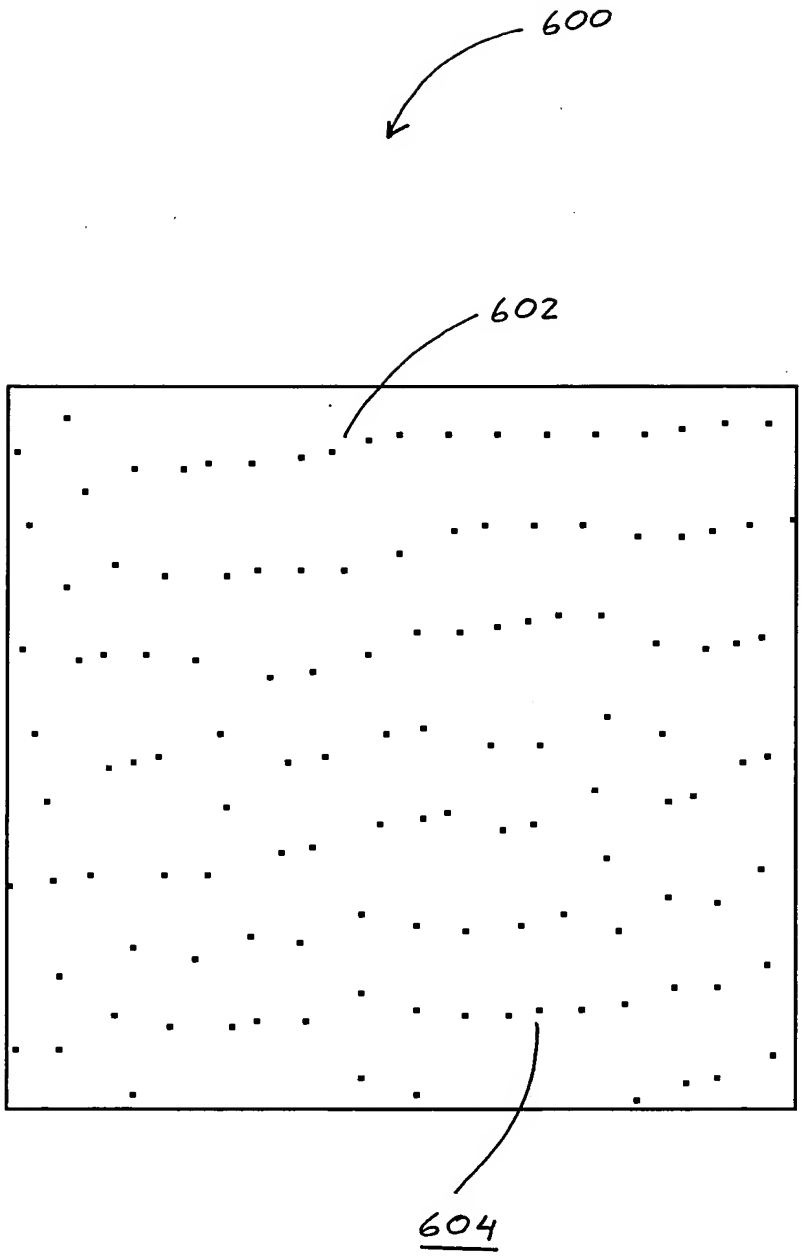

Fig. 6
(Prior Art)

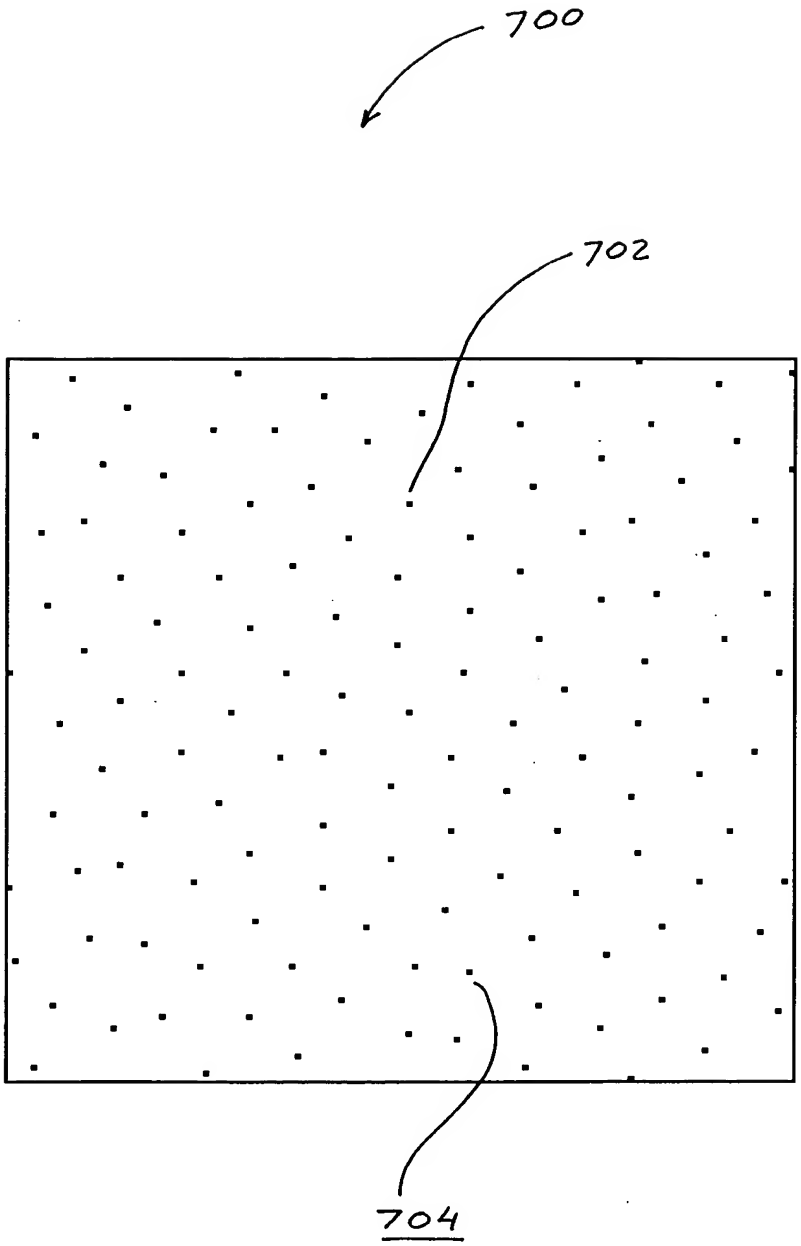

Fig. 7

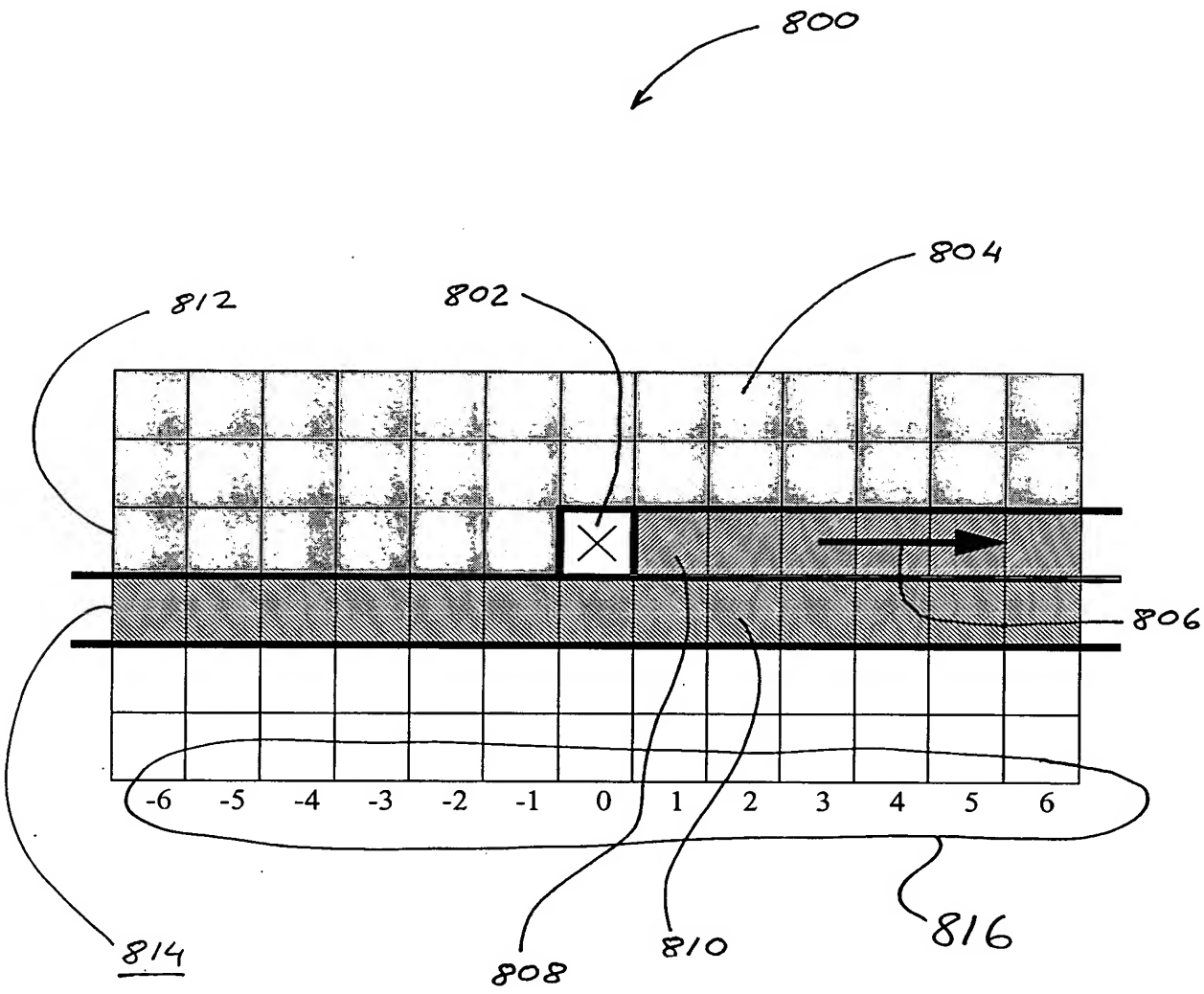

Fig. 8

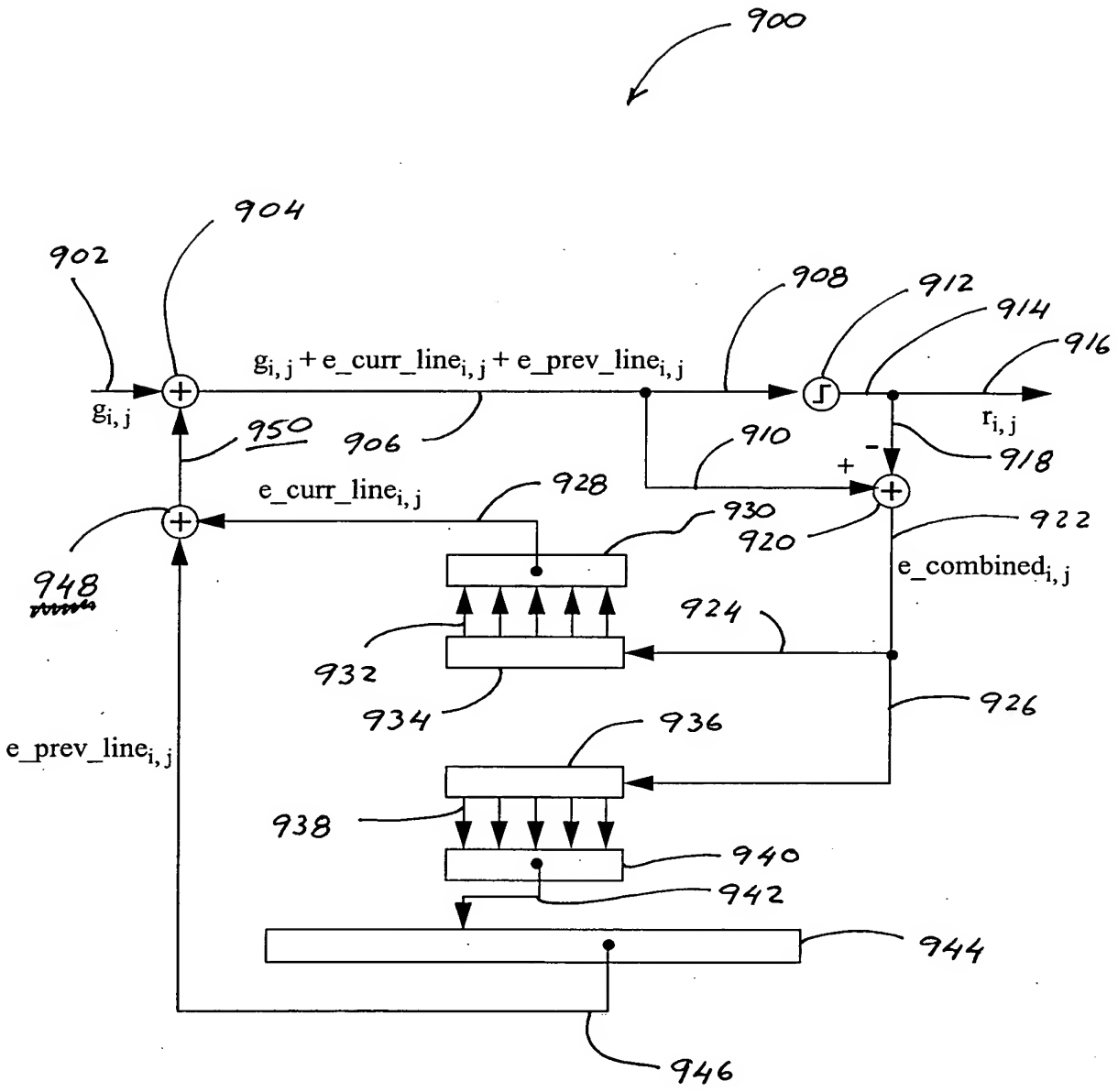
900
902
904
908
912
914
916
$g_{i,j} + e\_curr\_line_{i,j} + e\_prev\_line_{i,j}$
$g_{i,j}$
950
906
910
$r_{i,j}$
918
+
-
$e\_curr\_line_{i,j}$
928
930
920
922
$e\_combined_{i,j}$
948
924
932
934
926
$e\_prev\_line_{i,j}$
936
938
940
942
944
946

Fig. 9

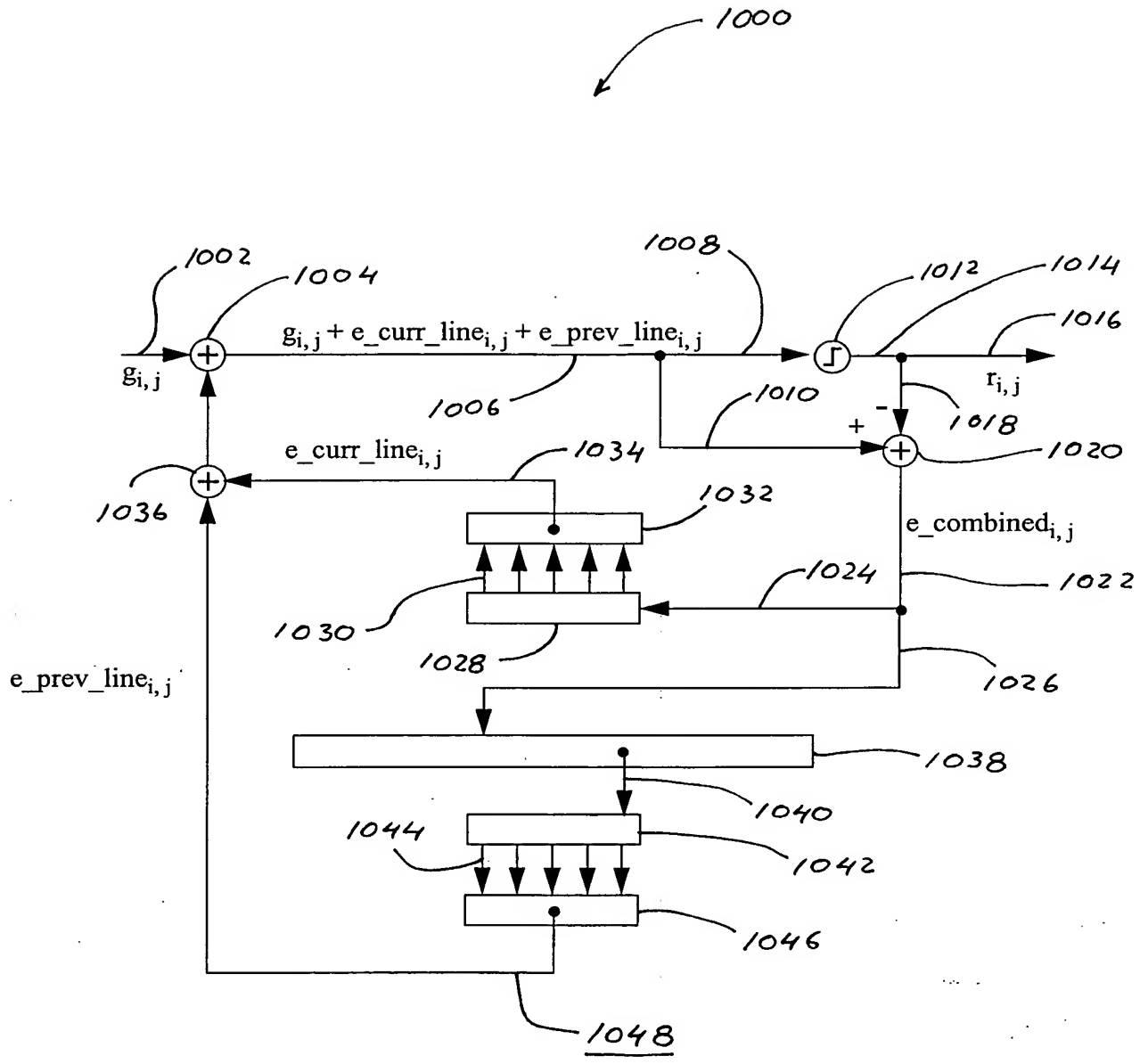
1000
1002
1004
1008
1012
1014
1016
$g_{i,j} + e\_curr\_line_{i,j} + e\_prev\_line_{i,j}$
$g_{i,j}$
1006
1010
$r_{i,j}$
1018
+
-
1020
$e\_curr\_line_{i,j}$
1034
1036
1032
$e\_combined_{i,j}$
1024
1022
1030
1028
$e\_prev\_line_{i,j}$
1026
1038
1040
1044
1042
1046
1048

Fig. 10

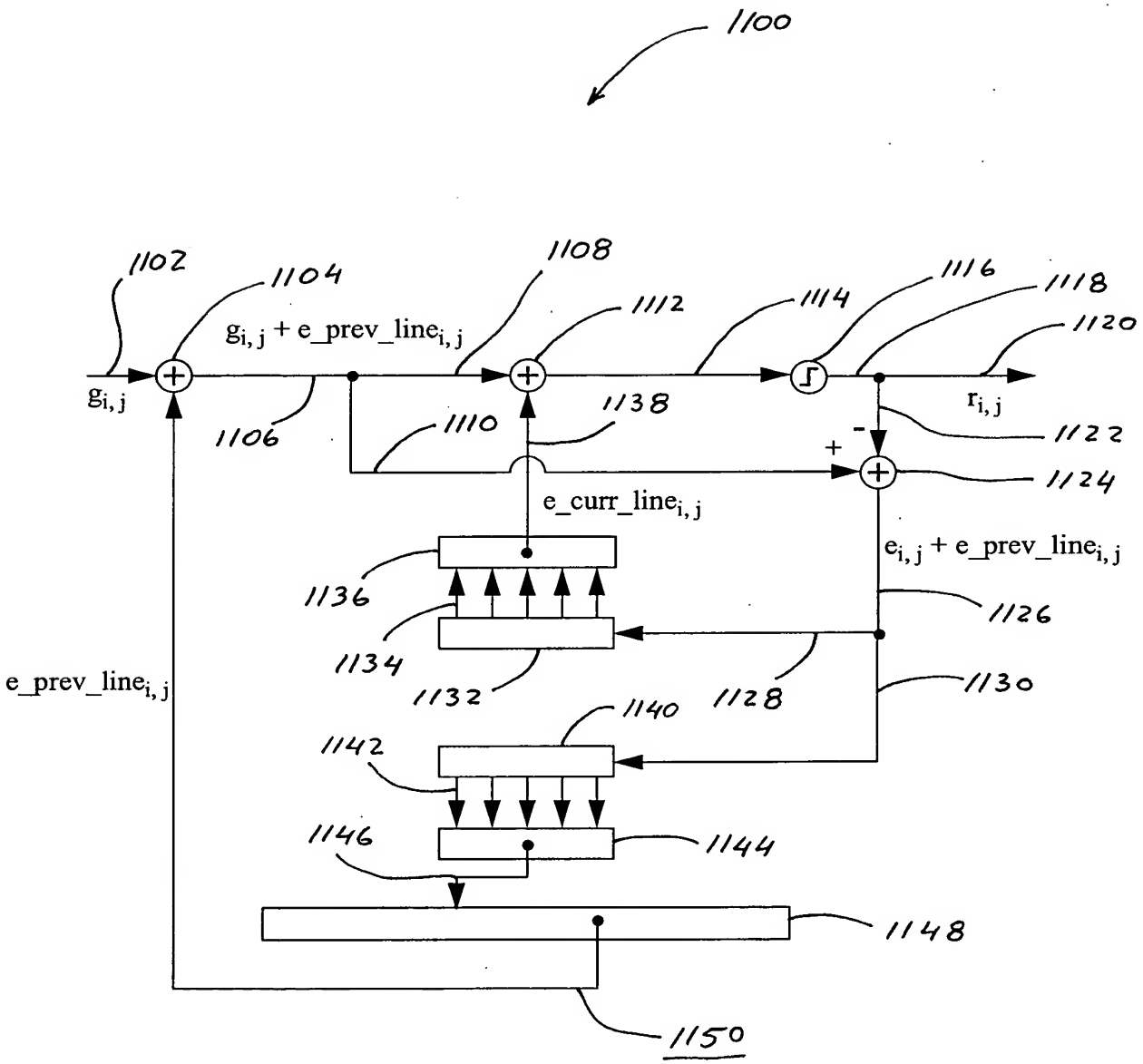
1100
1102
1104
1108
1112
1114
1116
1118
1120
$g_{i,j} + e\_prev\_line_{i,j}$
$g_{i,j}$
$r_{i,j}$
1106
1110
1138
1122
1124
$e\_curr\_line_{i,j}$
$e_{i,j} + e\_prev\_line_{i,j}$
1136
1126
1134
1132
1128
1130
$e\_prev\_line_{i,j}$
1140
1142
1146
1144
1148
1150

Fig. 11

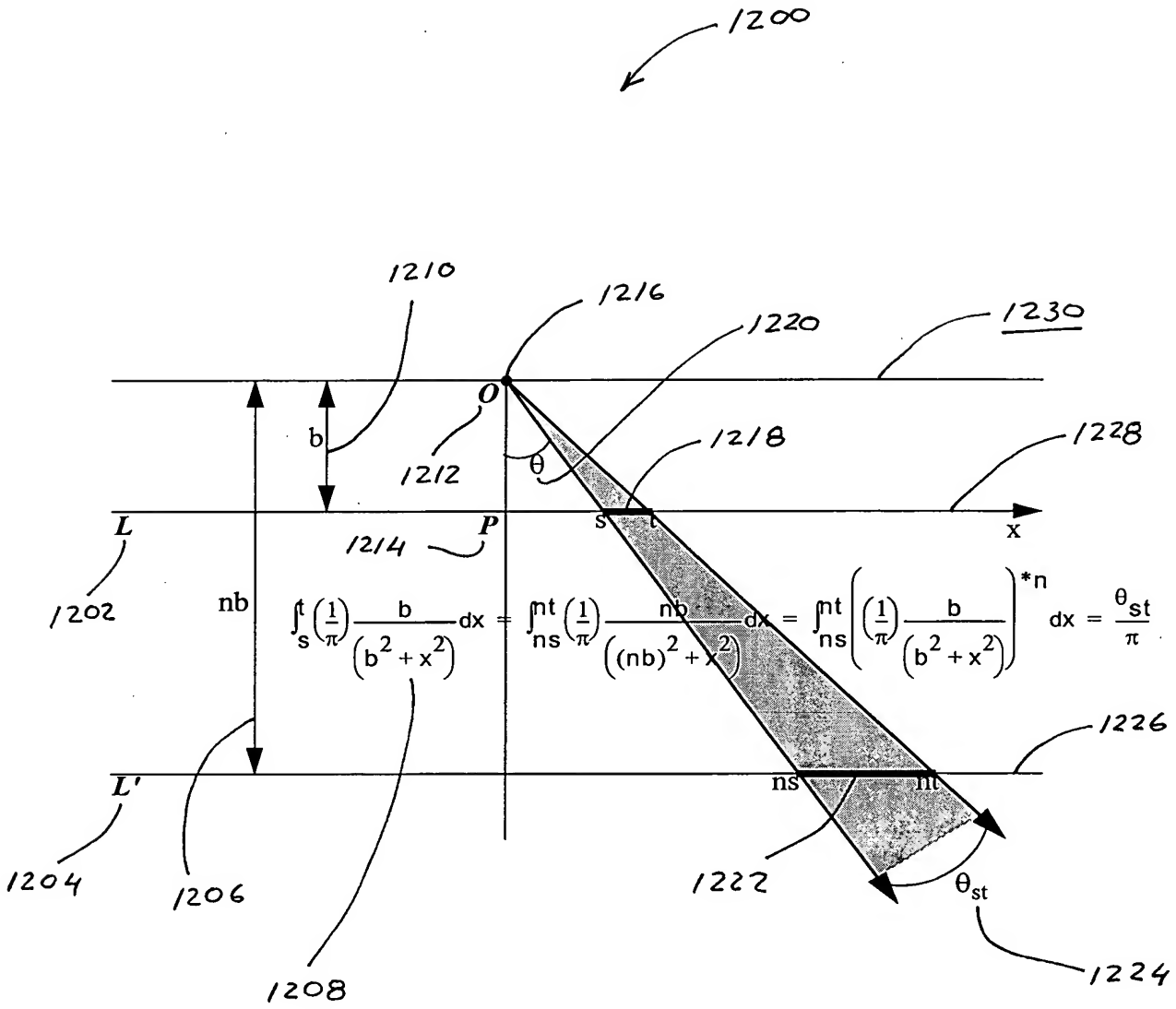
1200
1210
1216
1220
1230
1218
1228
1212
θ
b
O
L
1214
P
s
t
x
1202
nb
$\int_s^t \left(\frac{1}{\pi}\right)\frac{b}{\left(b^2+x^2\right)}dx = \int_{ns}^{nt}\left(\frac{1}{\pi}\right)\frac{nb}{\left((nb)^2+x^2\right)}dx = \int_{ns}^{nt}\left(\left(\frac{1}{\pi}\right)\frac{b}{\left(b^2+x^2\right)}\right)^{*n}dx = \frac{\theta_{st}}{\pi}$
1226
L'
ns
nt
1204
1206
1222
θ_{st}
1208
1224

Fig. 12

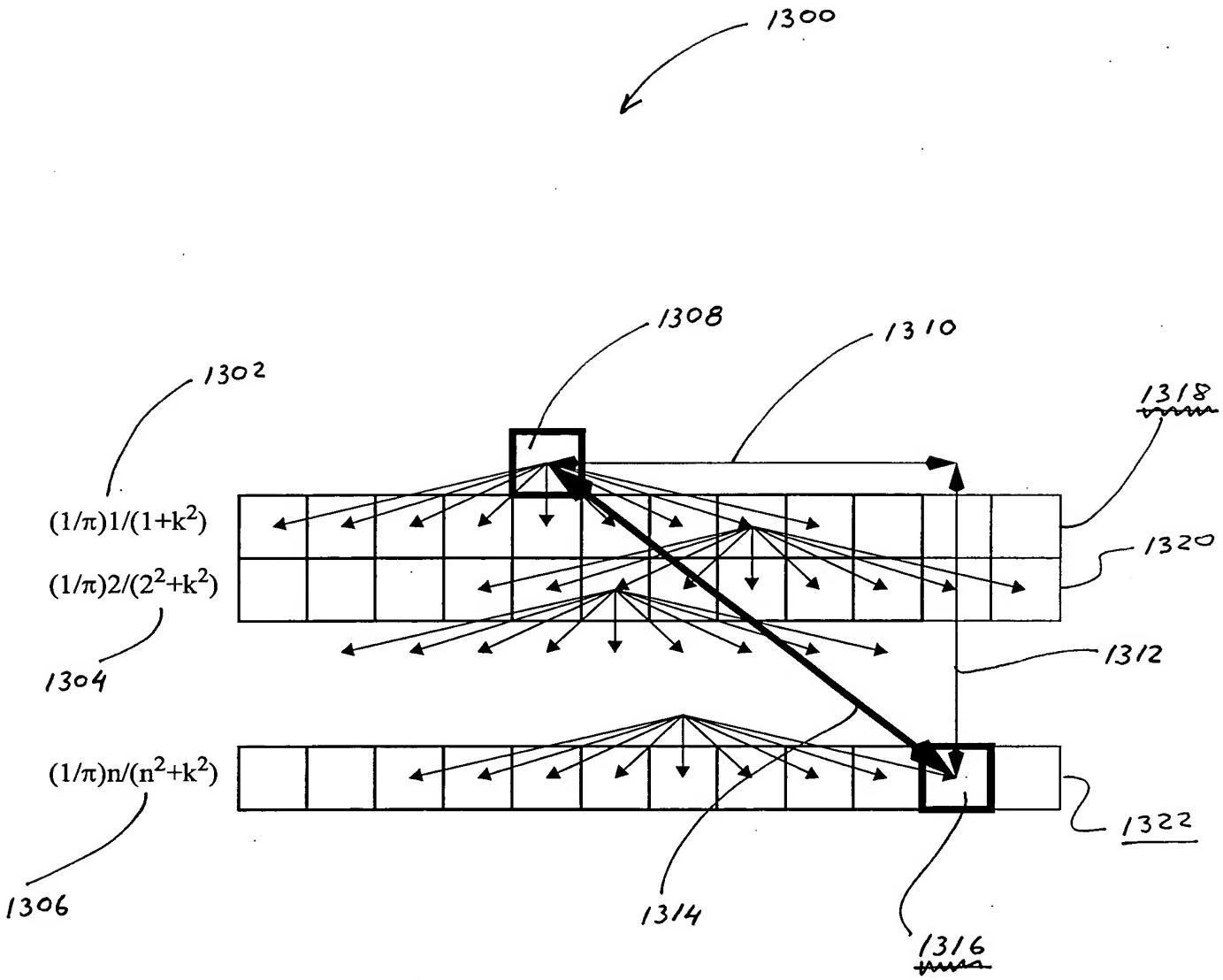
1300
1308
1310
1302
1318
$(1/\pi)1/(1+k^2)$
1320
$(1/\pi)2/(2^2+k^2)$
1312
1304
$(1/\pi)n/(n^2+k^2)$
1322
1306
1314
1316

Fig. 13

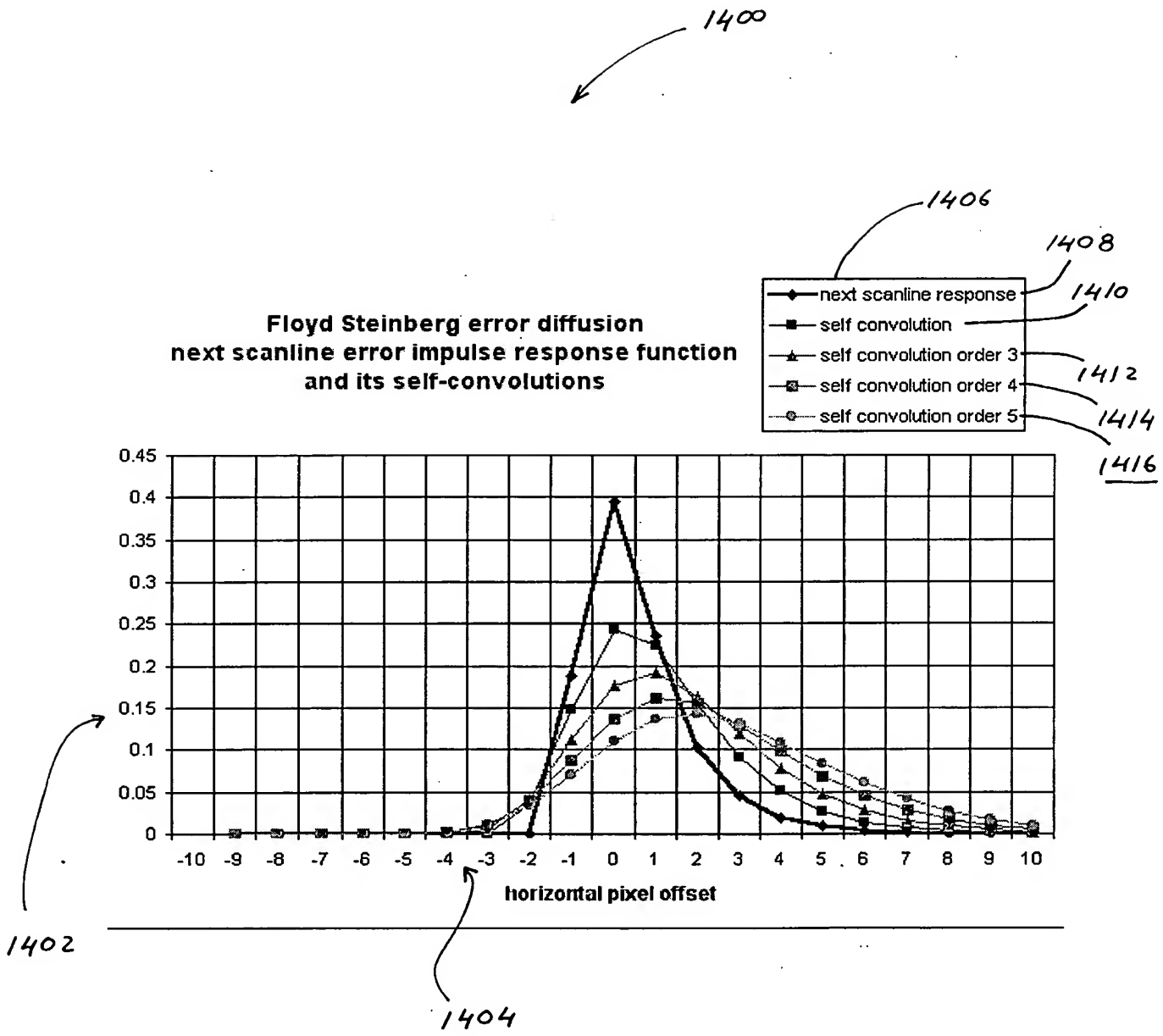
1400
1406
1408
1410
1412
1414
1416
next scanline response
self convolution
self convolution order 3
self convolution order 4
self convolution order 5
Floyd Steinberg error diffusion
next scanline error impulse response function
and its self-convolutions
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
horizontal pixel offset
1402
1404

Fig. 14

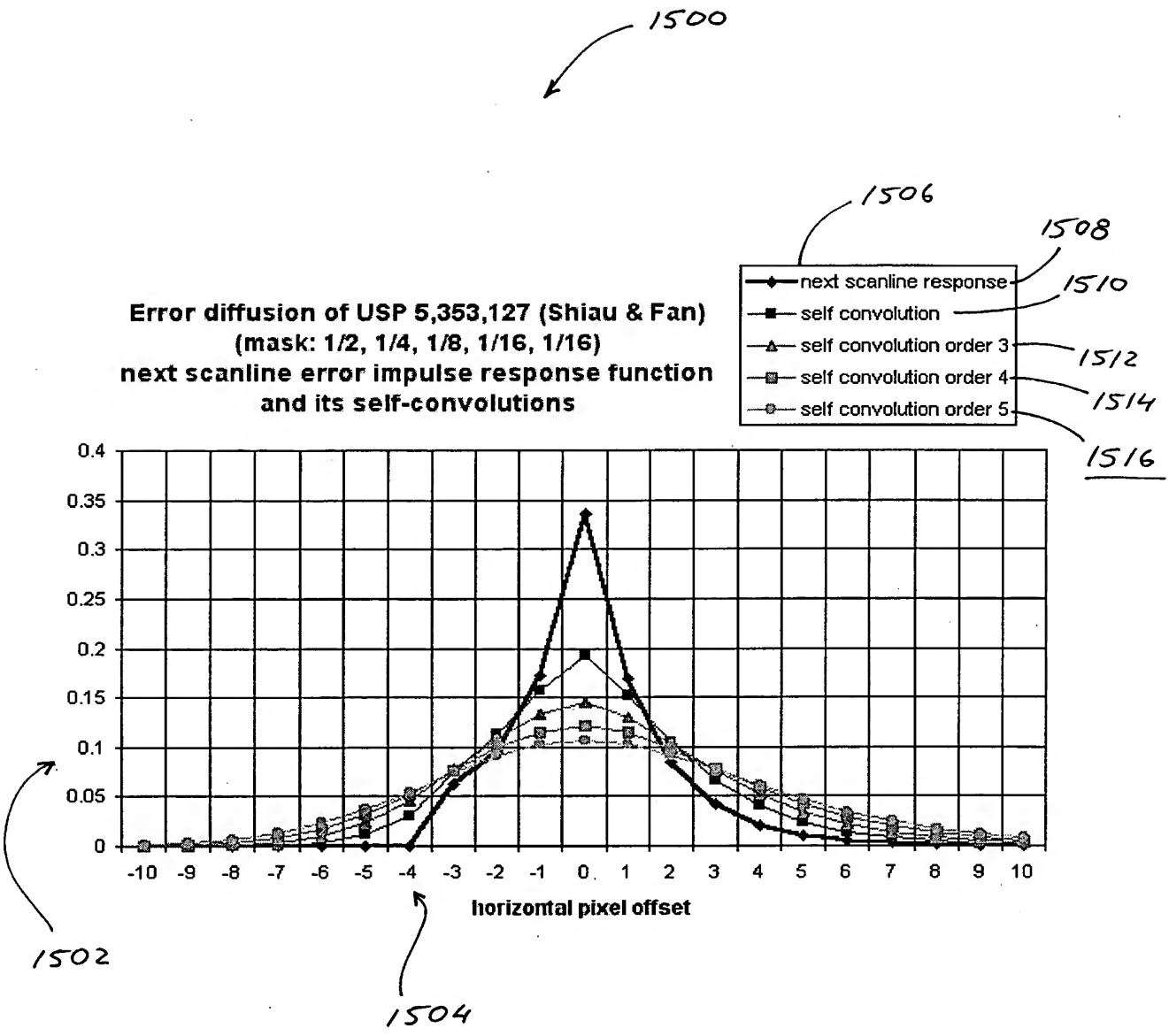
1500
1506
1508
1510
1512
1514
1516
Error diffusion of USP 5,353,127 (Shiau & Fan)
(mask: 1/2, 1/4, 1/8, 1/16, 1/16)
next scanline error impulse response function
and its self-convolutions
next scanline response
self convolution
self convolution order 3
self convolution order 4
self convolution order 5
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
horizontal pixel offset
1502
1504

Fig. 15

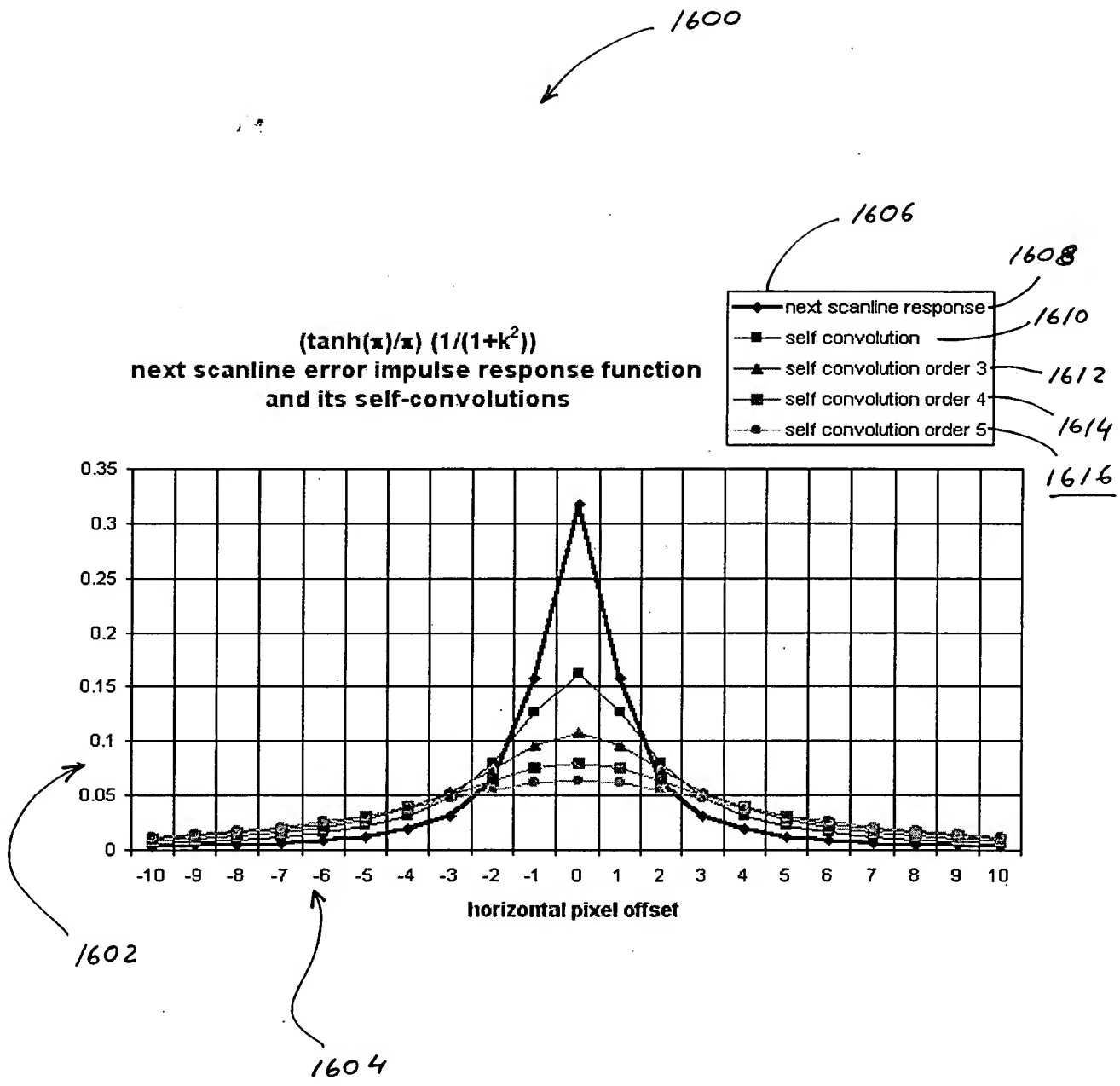
1600
1606
1608
1610
1612
1614
1616
next scanline response
self convolution
self convolution order 3
self convolution order 4
self convolution order 5
(tanh(x)/x) (1/(1+k²))
next scanline error impulse response function
and its self-convolutions
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
horizontal pixel offset
1602
1604

Fig. 16

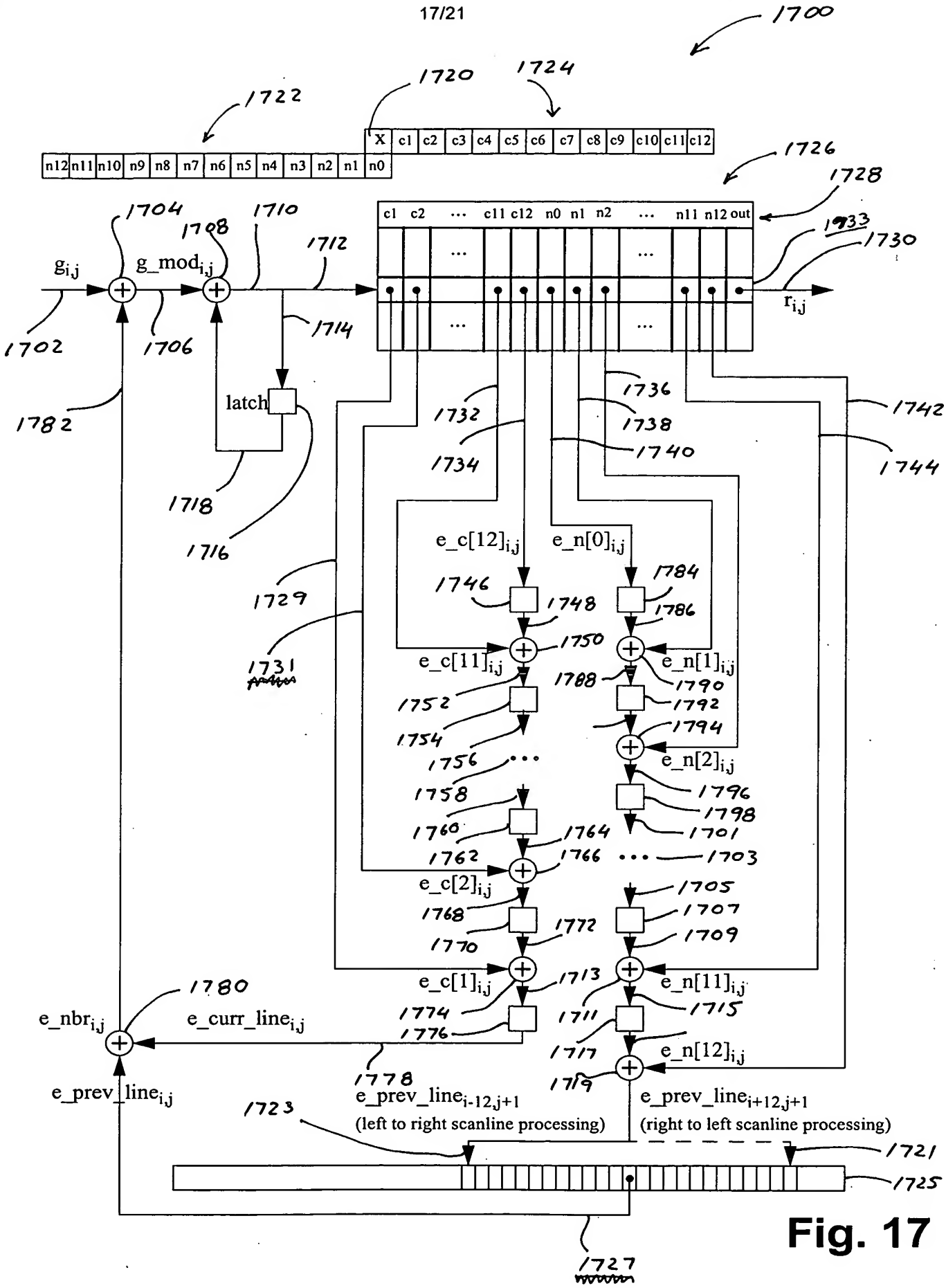
1700
1720
1724
1722
X c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12
n12 n11 n10 n9 n8 n7 n6 n5 n4 n3 n2 n1 n0
1726
1728
c1 c2 ... c11 c12 n0 n1 n2 ... n11 n12 out
1733
1730
1704
1708
1710
1712
$g_{i,j}$
$g\_mod_{i,j}$
$r_{i,j}$
1702
1706
1714
1782
latch
1718
1716
1732
1734
1736
1738
1740
1742
1744
1729
1731
$e\_c[12]_{i,j}$
$e\_n[0]_{i,j}$
1746
1748
1784
1786
1750
$e\_c[11]_{i,j}$
$e\_n[1]_{i,j}$
1752
1754
1788
1790
1792
1794
1756
$e\_n[2]_{i,j}$
1758
1796
1760
1798
1764
1701
1762
1766
1703
$e\_c[2]_{i,j}$
1705
1768
1707
1772
1770
1709
$e\_c[1]_{i,j}$
1713
$e\_n[11]_{i,j}$
1774
1711
1715
1776
1717
$e\_n[12]_{i,j}$
1780
$e\_nbr_{i,j}$
$e\_curr\_line_{i,j}$
1778
1719
$e\_prev\_line_{i,j}$
$e\_prev\_line_{i-12,j+1}$
(left to right scanline processing)
$e\_prev\_line_{i+12,j+1}$
(right to left scanline processing)
1723
1721
1725
1727

Fig. 17

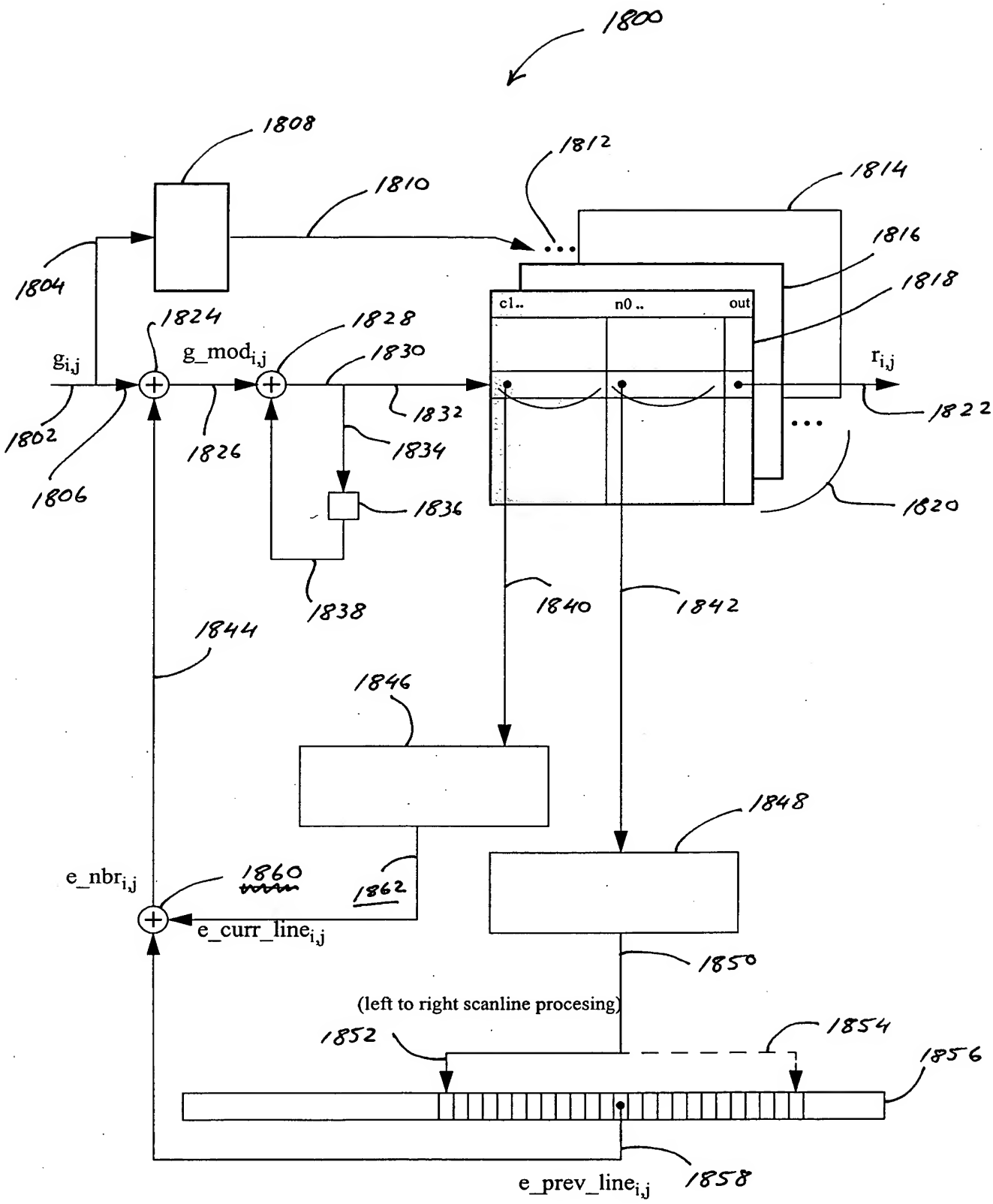
1800
1808
1812
1814
1810
1816
1804
1818
c1..
n0..
out
1824
1828
$g_{i,j}$
$g\_mod_{i,j}$
1830
$r_{i,j}$
1832
1822
1802
1826
1834
1806
1836
1820
1838
1840
1842
1844
1846
1848
$e\_nbr_{i,j}$
1860
1862
$e\_curr\_line_{i,j}$
1850
(left to right scanline procesing)
1852
1854
1856
$e\_prev\_line_{i,j}$
1858

Fig. 18

1900

pixel offset	mask 1	mask 2	mask 3	mask 4	mask 5	mask 6	mask 7	mask 8	mask 9	mask 10	mask 11	mask 12
c1	0.590522	0.474801	0.561741	0.536627	0.538362	0.536667	0.53547	0.535275	0.535142	0.535113	0.535478	0.534915
c2		0.063092	-0.122958	-0.059029	-0.080649	-0.076928	-0.07781	-0.077282	-0.080266	-0.080382	-0.08105	-0.080892
c3			0.099337	-0.006168	0.043546	0.02488	0.028431	0.026191	0.029559	0.029124	0.028874	0.0286
c4				0.057792	-0.01825	0.022132	0.008507	0.010402	0.008627	0.007733	0.007593	0.007557
c5					0.042134	-0.018664	0.010482	0.001604	0.002612	0.003383	0.003171	0.003258
c6						0.034487	-0.010039	0.012859	0.004904	0.005821	0.006146	0.005601
c7							0.025736	-0.010894	0.008603	0.001705	0.00219	0.002344
c8								0.021552	-0.008178	0.007955	0.002221	0.002784
c9									0.017524	-0.006994	0.00679	0.00211
c10										0.01448	-0.006819	0.005269
c11											0.012944	-0.005752
c12												0.011165
n0	0.222974	0.243788	0.236484	0.237737	0.237758	0.237552	0.237517	0.237154	0.237201	0.23708	0.236933	0.236962
n1	0.186504	0.137512	0.132786	0.130425	0.129196	0.128519	0.128013	0.127828	0.127623	0.127408	0.127224	0.127126
n2		0.080808	0.050122	0.051474	0.050109	0.049341	0.049103	0.048672	0.04855	0.04837	0.048253	0.048164
n3			0.042489	0.024984	0.025532	0.024637	0.023997	0.023767	0.023463	0.023307	0.02315	0.023072
n4				0.026159	0.014588	0.014971	0.014323	0.013879	0.013715	0.013511	0.013347	0.013238
n5					0.017672	0.009629	0.009888	0.009369	0.009058	0.0089	0.008756	0.008605
n6						0.012776	0.006755	0.00706	0.006642	0.006382	0.006227	0.006122
n7							0.009627	0.004983	0.005251	0.004924	0.004755	0.004631
n8								0.007582	0.003865	0.004091	0.003857	0.003692
n9									0.006104	0.003066	0.003254	0.003043
n10										0.005022	0.002511	0.002717
n11											0.004195	0.002068
n12												0.003602

1902 1904 1906 1908 1910 1912 1914 1916 1918 1920 1922 1924 1926

Fig. 19

mask index	grey levels
1	31-116, 138-224
2	28-30, 225-227, 117, 138
3	25-27, 228-230, 118, 137
4	22-24, 231-233, 119, 136
5	19-21, 234-236, 120, 135
6	16-18, 237-239, 121, 134
7	13-15, 240-242, 122, 133
8	10-12, 243-245, 123, 132
9	7-9, 246-248, 124, 131
10	4-6, 249-251, 125, 130
11	2-3, 252-253, 126, 129
12	0-1, 254-255, 127, 128

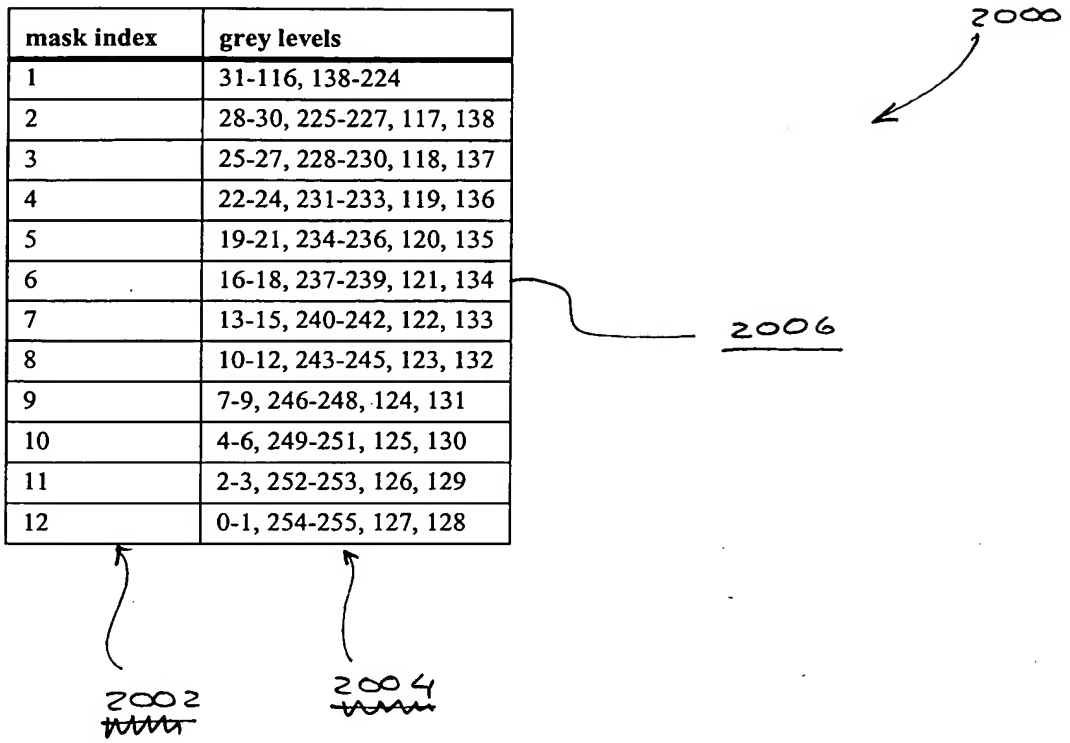

Fig. 20

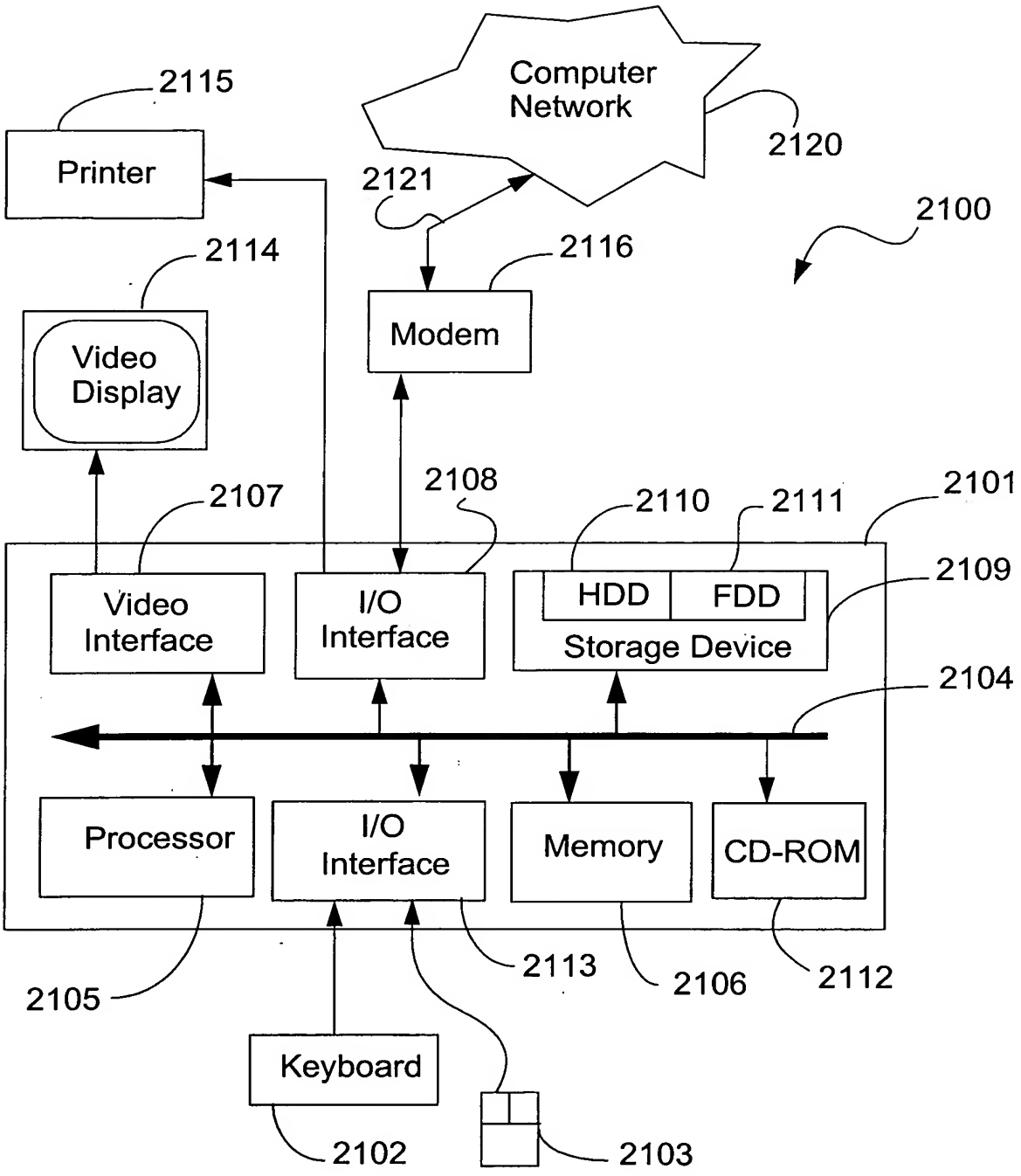
Computer Network
2120
2115
Printer
2121
2100
2114
2116
Video Display
Modem
2108
2107
2110
2111
2101
Video Interface
I/O Interface
HDD
FDD
Storage Device
2109
2104
Processor
I/O Interface
Memory
CD-ROM
2105
2113
2106
2112
Keyboard
2102
2103

Fig. 21